ENKBB-2.0

VAX 11/730 8085
TEST PART II

EP-ENKBB-DL-2.0
FICHE 1 OF 1

JAN 1983
COPYRIGHT© 82-83
MADE IN USA

IDENTIFICATION

Product code: ENKBB VERSION 2.0

Product name: 8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS AND CPU MODULES

Product date: 11-OCT-82

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC DECsystem-10 DECSYSTEM-20
DECUS MASSBUS PDP
UNIBUS VAX VMS

digital

Table of Contents

1.0 ABSTRACT 3

2.0 HARDWARE REQUIREMENTS 3

3.0 SOFTWARE REQUIREMENTS 3

4.0 PREREQUISITES 3

5.0 OPERATING INSTRUCTIONS 3
5.1 Options 3
5.2 Event Flags 4
5.3 APT Setup 4

6.0 PROGRAM FUNCTIONAL DESCRIPTION 4
6.1 Program Overview 4
6.2 Program Size 4
6.3 Program Run Times 4
6.4 Run-time Dynamics 4
6.5 Fault Detection 5
6.6 Performance During Hardware Failures 5
6.7 Program Applications 5
6.8 Test Descriptions 6

7.0 MAINTENANCE HISTORY 6

1.0 ABSTRACT

This program is an 8085 based micro-diagnostic designed to test the hardware on the WCS (Writable Control Store) and DAP (Data Path) modules of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with a minimum hardware configuration of the WCS and CPU modules present.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

4.0 PREREQUISITES

This module should be run after running the module ENKBA.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKBB after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following descibes the APT parameters needed to execute this diagnostic:
TBS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 3K bytes long.

6.3 Program Run Times

This program will take approximately 1 minute and 40 seconds to run.

6.4 Run-time Dynamics

Not applicable

6.5 Fault Detection

The diagnostic will report errors with the following header:

```
SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

1. SECT is the program name (ENKBB)
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is APT compatable.

Customers may use the program to verify the basic integrity of the WCS and CPU modules in the system.

6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
11-OCT-82	2.0	RD changes.

```
;       ENKBB  8085 based diagnostic for WCS/DAP module  Rev 02.00
;       Table of Contents
;
;  135  DATA AREA
;  302  PROGRAM ENTRY POINT
;  322  VARIABLES AREA
;  471  PROGRAM SPECIFIC SUBROUTINES
;  472    CON_SET_FOR_CO
;  489    CHECK_RD
;  510    LD_SUPPORT
;  567    SET_COMPARE
;  593    LS_WRITE
;  623    LS_READ
;  653    FAST_WRITE_WRO
;  703    FAST_READ_WRO
;  732    READ_BYTE_WRO
;  762    WRITE_WR1_32
;  800    READ_WR1_32
;  842    SHIFT_WRT_R8
;  869    EXEC_INST
;  875    LD_CSR_INST
;  916    LOAD_MOD_DAT
;  930    TEST9_1_0
;  963    TEST9_SHIFT
; 1001    TEST9_2_0
; 1047    TEST9_ERROR
; 1050    TEST9_ERR2_3
; 1074    TESTA_ERROR
; 1100    TESTB_ERROR
; 1121    TESTD_ERROR
; 1124    TESTC_ERROR
; 1147    TESTE_F_ERROR
; 1175    REPORT_CPU_ERROR
; 1179    REPORT_ERROR
; 1238  TEST 9 - 2901 DATA PATH TEST (WORKING REG 0 AND WR CRR TESTS)
; 1398  TEST A - 2901 SHIFT FUNCTION TEST
; 1569  TEST B - WORKING REGISTER CLEAR TEST
; 1881  TEST C - WORKING REGISTER TEST (WORKING REG 1)
; 2152  TEST D - 32 BIT D AND Y BUSES TEST
; 2399  TEST E - LS RAM DATA TEST
; 2535  TEST F - LS RAM MARCH PATTERN TEST
```

```
.TITLE  "ENKBB  8085 based diagnostic for WCS/DAP module  Rev 02.00"
;
;
;                           COPYRIGHT (c) 1982 BY
;                  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
;                   MASSACHUSETTS.  ALL RIGHTS RESERVED.
;
; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
;
; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
;
; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
;
;
;++
; FACILITY:      VAX-11/730 MICRO-DIAGNOSTIC.
;
; ABSTRACT:      This 8085 based diagnostic tests the hardcore logic of the
;                WCS and DAP modules.
;
; ENVIRONMENT:   ENKAA  Micro-diagnostic Monitor
;
; AUTHOR:        Jim Klumpp      4-JUN-81
;
; MODIFIED BY:   David Mayo      8-MAR-1982       VERSION 01.00
;                Ernie Preisig  26-Aug-1982       Version 02.00   RD Changes
;
;--
```

```
7000   128
7000   129
7000   130
7000   131
7000   132
7000   133
7000   134
7000   135
7000   136 000                                                 .PSECT  CPU2,BYTE
7000   137                                                     .ALIGN  BYTE
7000   138
7000   139                                      ;*************************************************************************
7000   140                                      ;
7000   141                                      ; DATA AREA
7000   142                                      ;
7000   143                                      ;*************************************************************************
7000   144
7000   145 00000000                             HEAD           =0
7000   146
7000   147 00000000                                            EQUATE                  ;REGISTER EQUATE MACRO
           00000002
           00000004
           00000006
           00000006
           00000001
           00000003
           00000005
           00000007 00000006
7000   148
7000   149 00000000                             UPC14A         =<^X00>                 ;(RO) UPC BIT 14 ASSERTED HIGH <MSB>
7000   150
7000   151
7000   152 00000001                             MISC2          =<^X01>
7000   153 00000001                             BOOTF          =<^X01>                 ;(RO) BOOT FLAG. ASSERTED LOW. <LSB>
7000   154
7000   155
7000   156 00000002                             DCLO           =<^X02>                 ;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
7000   157
7000   158 00000002                             HLTFLG         =<^X02>                 ;(RO) HALT FLAG ASSERTED LOW    <LSB>
7000   159
7000   160
7000   161 00000003                             READJ1         =<^X03>                 ;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
7000   162
7000   163 00000003                             ALLOWP         =<^X03>                 ;(RO) ENABLE CONTROL P ASS LOW <LSB>
7000   164
7000   165
7000   166 00000004                             READJ2         =<^X04>                 ;(RO) BIT 07 EQUALS J2 <MSB>
7000   167 00000004                             APTFLG         =<^X04>                 ;(RO) APT PRESENT FLAG <LSB>
7000   168
7000   169 00000005                             AUTO           =<^X05>                 ;(RO) AUTO TEST FLAG ASS LOW <LSB>
7000   170
7000   171 00000006                             REMDIS         =<^X06>                 ;(RO) REMOTE DISABLE FLAG ASS LOW <LSB>
7000   172
7000   173 00000007                             OKV15          =<^X07>                 ;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
7000   174
7000   175 00000007                             BOOTEN         =<^X07>                 ;(RO) BOOT ENABLE FLAG ASS LOW <LSB>
7000   176
7000   177
7000   178 00000008                             WCSB0          =<^X08>                 ;(WO) WCS WRITE DATA REGISTER
7000   179 00000009                             WCSB1          =<^X09>
```

```
7000   180 0000000A                     WCSB2           =<^X0A>
7000   181
7000   182 0000001C                     ITDB            =<^X1C>                 ;(R/W) TIMER DATA REGISTER
7000   183 0000001D                     ITCSR           =<^X1D>                 ;(R/W) TIMER STATUS REGISTER
7000   184
7000   185 00000020                     CLRCLK          =<^X20>                 ;(WO) CLEAR CPU CLOCK RUN
7000   186 00000021                     SETCLK          =<^X21>                 ;(WO) SET CPU CLOCK RUN
7000   187
7000   188 00000022                     CLRSS           =<^X22>                 ;(WO) CLEAR CLOCK SINGLE STEP
7000   189 00000023                     SETSS           =<^X23>                 ;(WO) SET CLOCK SINGLE STEP
7000   190
7000   191 00000024                     CLRCSK          =<^X24>                 ;(WO) CLEAR CSR CLOCK
7000   192 00000025                     SETCSK          =<^X25>                 ;(WO) SET CSR CLOCK
7000   193
7000   194 00000026                     CLRUPK          =<^X26>                 ;(WO) CLEAR THE UPC CLOCK
7000   195 00000027                     SETUPK          =<^X27>                 ;(WO) SET THE UPC CLOCK
7000   196
7000   197 00000028                     SETMCI          =<^X28>                 ;(WO) SET MEMORY CONTROL INIT
7000   198 00000029                     CLRMCI          =<^X29>                 ;(WO) CLEAR MEMORY CONTROL INIT
7000   199
7000   200 0000002A                     SETMCK          =<^X2A>                 ;(WO) SET MEMORY CONTROL CLOCK
7000   201 0000002B                     CLRMCK          =<^X2B>                 ;(WO) CLEAR MEMORY CONTROL CLOCK
7000   202
7000   203 0000002C                     CLRTMR          =<^X2C>                 ;(WO) STOP TIMER
7000   204 0000002D                     SETTMR          =<^X2D>                 ;(WO) START TIMER COUNTING
7000   205
7000   206 0000002E                     CLRBSY          =<^X2E>                 ;(WO) CLEAR UNIBUS BUS BUSY
7000   207 0000002F                     SETBSY          =<^X2F>                 ;(WO) SET UNIBUS BUS BUSY
7000   208
7000   209 00000030                     SETDCL          =<^X30>                 ;(WO) SET DC LOW
7000   210 00000031                     CLRDCL          =<^X31>                 ;(WO) CLEAR DC LOW
7000   211
7000   212 00000032                     SETACL          =<^X32>                 ;(WO) SET AC LOW
7000   213 00000033                     CLRACL          =<^X33>                 ;(WO) CLEAR AC LOW
7000   214
7000   215 00000034                     INITL           =<^X34>                 ;(WO) CLEAR UNIBUS INIT
7000   216 00000035                     INITH           =<^X35>                 ;(WO) SET UNIBUS INIT
7000   217
7000   218 00000036                     CLRWCK          =<^X36>                 ;(WO) CLEAR WCS BIT 00
7000   219 00000037                     SETWCK          =<^X37>                 ;(WO) SET WCS BIT 00
7000   220
7000   221 00000038                     CLRCSR          =<^X38>                 ;(WO) CLEAR WCS BIT 01
7000   222 00000039                     SETCSR          =<^X39>                 ;(WO) SET WCS BIT 01
7000   223
7000   224 0000003C                     CLRLIT          =<^X3C>                 ;(WO) CLEAR THE RUN LIGHT
7000   225 0000003D                     SETLIT          =<^X3D>                 ;(WO) SET THE RUN LIGHT
7000   226
7000   227 00000044                     TUDB            =<^X44>                 ;(R/W) TU-58 DATA BUFFER
7000   228 00000045                     TUSR            =<^X45>                 ;(RO)  TU-58 STATUS REGISTER
7000   229 00000046                     TUMODE          =<^X46>                 ;(R/W) TU-58 MODE REGISTER 1 + 2
7000   230                                                                      ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000   231                                                                      ;READING THE CR FORCES THE INTERNAL
7000   232                                                                      ;POINTER BACK TO MR1
7000   233 00000047                     TUCR            =<^X47>                 ;(R/W) TU-58 COMMAND REGISTER
7000   234
7000   235 00000048                     TTDB            =<^X48>                 ;(R/W) TERMINAL DATA BUFFER
7000   236 00000049                     TTSR            =<^X49>                 ;(RO)  TERMINAL STATUS REGISTER
7000   237 0000004A                     TTMODE          =<^X4A>                 ;(R/W) TERMINAL MODE REGISTER 1 + 2
7000   238                                                                      ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000   239                                                                      ;READING THE CR FORCES THE INTERNAL
```

```
7000  240                                                                    ;POINTER BACK TO MR1
7000  241 0000004B                            TTCR            =<^X4B>        ;(R/W) TERMINAL COMMAND REGISTER
7000  242
7000  243 00000080                            READ            =<^X80>        ;(RO) CONSOLE READ REGISTER
7000  244
7000  245 00000082                            CPUACK          =<^X82>        ;(RO) ACKNOWLEDGE FROM CPU        <LSB>
7000  246
7000  247 00000083                            CPATTN          =<^X83>        ;(RO) ATTENTION FROM CPU          <LSB>
7000  248
7000  249 00000084                            UPC14           =<^X84>        ;(RO) UPC BIT 14                  <LSB>
7000  250
7000  251 00000085                            CSR07           =<^X85>        ;(RO) CSR BIT 7                   <LSB>
7000  252
7000  253 00000086                            CSR15           =<^X86>        ;(RO) CSR BIT 15                  <LSB>
7000  254
7000  255 00000087                            CSR23           =<^X87>        ;(RO) CSR BIT 23                  <LSB>
7000  256
7000  257 000000A0                            ENBPAR          =<^XA0>        ;(WO) ENABLE STALL ON PARITY ERROR
7000  258 000000A1                            DISPAR          =<^XA1>        ;(WO) DISABLE STALL ON PARITY ERROR
7000  259
7000  260 000000A2                            VSHIFT          =<^XA2>        ;(WO) SET THE V-BUS TO SHIFT MODE
7000  261 000000A3                            VNORML          =<^XA3>        ;(WO) SET THE V-BUS TO NORMAL MODE
7000  262
7000  263 000000A4                            SETTIM          =<^XA4>        ;(WO) SET THE INTERVAL TIMER INTERRUPT
7000  264 000000A5                            CLRTIM          =<^XA5>        ;(WO) CLEAR THE INTERVAL TIMER INTRPT.
7000  265
7000  266 000000A6                            ENBMEM          =<^XA6>        ;(WO) ENABLE MEMORY REFERENCES
7000  267 000000A7                            DISMEM          =<^XA7>        ;(WO) DISABLE MEMORY REFERENCES
7000  268
7000  269 000000A8                            SETACK          =<^XA8>        ;(WO) SET CONSOLE ACKNOWLEDGE
7000  270 000000A9                            CLRACK          =<^XA9>        ;(WO) CLEAR CONSOLE ACKNOWLEDGE
7000  271
7000  272                                     ;NOTE:  THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC
7000  273                                     ;SERIAL INPUT.  WHEN USED THIS WAY THE SENSE IS INVERTED.  TO SIMPLIFY
7000  274                                     ;MATTERS THE NEXT TWO EQUATES ARE INCLUDED
7000  275
7000  276 000000A8                            CLRUPC          =<^XA8>        ;(WO) CLEAR THE V-BUS SERIAL INPUT
7000  277 000000A9                            SETUPC          =<^XA9>        ;(WO) SET THE V-BUS SERIAL INPUT
7000  278
7000  279 000000AA                            SETATN          =<^XAA>        ;(WO) SET CONSOLE ATTENTION
7000  280 000000AB                            CLRATN          =<^XAB>        ;(WO) CLEAR CONSOLE ATTENTION
7000  281
7000  282 000000AC                            SETPFI          =<^XAC>        ;(WO) SET POWER FAIL INTERRUPT
7000  283 000000AD                            CLRPFI          =<^XAD>        ;(WO) CLEAR POWER FAIL INTERRUPT
7000  284
7000  285 000000AE                            SETHLT          =<^XAE>        ;(WO) SET THE CONSOLE HALT BIT
7000  286 000000AF                            CLRHLT          =<^XAF>        ;(WO) CLEAR THE CONSOLE HALT BIT
7000  287
7000  288 000000CC                            WRITE           =<^XCC>        ;(WO) THE CONSOLE WRITE REGISTER
7000  289
7000  290 000000EC                            YBUSRD          =<^XEC>        ;(RO) Y-BUS READ
7000  291
7000  292
7000  293
7000  294                                     ;ADDITIONAL DATA
7000  295
7000  296 00004C22                            PARITY_VECTOR   = <^X4C22>     ;PARITY ERROR BRANCH VECTOR.
7000  297 00004C25                            POWER_VECTOR    = <^X4C25>     ;POWER FAIL BRANCH VECTOR.
7000  298 000040D9                            MIPFLG          =<^X40D9>      ;ADDRESS OF MODEM IN PROGRESS FLAG
7000  299
```

```
7000  300
7000  301                                 ;******************************************************************************
7000  302                                 ;
7000  303                                 ;  PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
7000  304                                 ;
7000  305                                 ;******************************************************************************
7000  306
7000  307 0450'C3                         ENTRY:          JMP     TEST9
7003  308
7003  309                                 ;******************************************************************************
7003  310                                 ;
7003  311                                 ;  VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
7003  312                                 ;  BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
7003  313                                 ;
7003  314                                 ;******************************************************************************
7003  315
7003  316 0200                            VERSION:        .WORD           <^X0200>        ;VERSION NUMBER (REV 02.00)
7005  317 42 42                           FILE_NAME:      .ASCII          'BB''           ;LAST 2 CHARS OF FILE NAME
7007  318
7007  319                                 ;******************************************************************************
7007  320                                 ;
7007  321                                 ; VARIABLES AREA
7007  322                                 ;
7007  323                                 ;******************************************************************************
7007  324
7007  325
7007  326 00000008                        SHIFT_CNT:      .BLKB           1               ;COUNTER FOR WRK REG SHIFTS.
7008  327 04                              SHIFT_CNT_DAT:  .BYTE           4               ;COUNTER DATA.
7009  328 00                              LS_ADDR:        .BYTE           0               ;LS ADDRESS POINTER.
700A  329 0000                            DATA_PNT:       .WORD           0               ;TEST E DATA POINTER.
700C  330 00                              LOOP_CNT:       .BYTE           0               ;COUNT FOR ^C USE.
700D  331 00                              SAVE_LS_ADDR:   .BYTE           0               ;SAVES LS POINTER ON ^C.
700E  332
700E  333 37                              MOV_CCR_WR0:    .BYTE           <^X37>          ;INSTRUCTION USED TO PASS DATA
700F  334 F8                                              .BYTE           <^XF8>          ; FROM THE D-BUS TO WR[0].
7010  335 15                                              .BYTE           <^X15>
7011  336
7011  337 BE                              MOV_WR0_LS0:    .BYTE           <^XBE>          ;INSTRUCTION USED TO PASS DATA
7012  338 00                                              .BYTE           <^X00>          ; FORM WR[0] TO LS[0].
7013  339 15                                              .BYTE           <^X15>
7014  340
7014  341 36                              MOV_LS0_WR0:    .BYTE           <^X36>          ;INSTRUCTION USED TU PASS DATA
7015  342 00                                              .BYTE           <^X00>          ; FORM LS[0] TO WR[0].
7016  343 15                                              .BYTE           <^X15>
7017  344
7017  345 BE                              MOV_WR0_LSX:    .BYTE           <^XBE>          ;INSTRUCTION USED TO PASS DATA
7018  346 00                                              .BYTE           <^X00>          ; FROM WR[0] TO AN LS LOCATION
7019  347 15                                              .BYTE           <^X15>          ; SPECIFIED DURING EXECUTION.
701A  348
701A  349 36                              MOV_LSX_WR0:    .BYTE           <^X36>          ;INSTRUCTION USED TO PASS DATA
701B  350 00                                              .BYTE           <^X00>          ; FROM AN LS LOCATION SPECIFIED
701C  351 15                                              .BYTE           <^X15>          ; DURING EXECUTION TO WR[0].
701D  352
701D  353 A0                              MOV_WR1_WR1:    .BYTE           <^XA0>          ;INSTRUCTION TO PASS DATA FROM
701E  354 02                                              .BYTE           <^X02>          ; WR[1] TO THE Y-BUS.
701F  355 95                                              .BYTE           <^X95>
7020  356
7020  357 A3                              SHIFT_WR1_R:    .BYTE           <^XA3>          ;INSTRUCTION USED TO SHIFT THE
7021  358 40                                              .BYTE           <^X40>          ; WR[1] CONTENTS ONE PLACE TO
7022  359 95                                              .BYTE           <^X95>          ; THE RIGHT.
```

```
7023
7023  23          ROT_WR1_L:      .BYTE   <^X23>          ;INSTRUCTION USED TO SHIFT THE
7024  C0                          .BYTE   <^XC0>          ; WR[1] CONTENTS ONE PLACE TO
7025  95                          .BYTE   <^X95>          ; THE LEFT WITH LSB SET.
7026
7026  A3          SHIFT_WR1_L:    .BYTE   <^XA3>          ;INSTRUCTION USED TO SHIFT THE
7027  D0                          .BYTE   <^XD0>          ; WR[1] CONTENTS ONE PLACE TO
7028  95                          .BYTE   <^X95>          ; THE LEFT WITH LSB CLEAR.
7029
7029  2F          X_CLR_WR1:      .BYTE   <^X2F>          ;INSTRUCTION USED TO CLEAR
702A  82                          .BYTE   <^X82>          ; WR[1].
702B  95                          .BYTE   <^X95>
702C
702C  A0          X_COM_WR1:      .BYTE   <^XA0>          ;INSTRUCTION USED TO COMPLEMENT
702D  C2                          .BYTE   <^XC2>          ; WR[1].
702E  95                          .BYTE   <^X95>
702F
702F  10          MISC2_WR_CRR:   .BYTE   <^X10>          ;INSTRUCTION USED TO ENABLE
7030  FE                          .BYTE   <^XFE>          ; WR CRR
7031  15                          .BYTE   <^X15>
7032
7032  2F          SUPPORT_CODE:   .BYTE   <^X2F>          ;SIX WCS MICROINSTRUCTIONS USED
7033  80                          .BYTE   <^X80>          ; FOR FAST WRITES OF WCS.
7034  15                          .BYTE   <^X15>
7035
7035  A0                          .BYTE   <^XA0>          ;*
7036  C0                          .BYTE   <^XC0>
7037  15                          .BYTE   <^X15>
7038
7038  5B                          .BYTE   <^X5B>          ;*
7039  00                          .BYTE   <^X00>
703A  18                          .BYTE   <^X18>
703B
703B  A3                          .BYTE   <^XA3>          ;*
703C  D0                          .BYTE   <^XD0>
703D  1E                          .BYTE   <^X1E>
703E
703E  A3                          .BYTE   <^XA3>          ;*
703F  C0                          .BYTE   <^XC0>
7040  15                          .BYTE   <^X15>
7041
7041  08                          .BYTE   <^X08>          ;*
7042  00                          .BYTE   <^X00>
7043  24                          .BYTE   <^X24>
7044
7044  0001        VER_WRITE_TAB:  .WORD   <^X0001>        ;*
7046  0002                        .WORD   <^X0002>        ;*
7048  0003                        .WORD   <^X0003>        ;EXPECTED UPC ADDRESSES IN
704A  0004                        .WORD   <^X0004>        ; STEPPING THROUGH THE FAST
704C  0006                        .WORD   <^X0006>        ; WRITE WRO MICROCODE.
704E  0003                        .WORD   <^X0003>        ;*
7050  0005                        .WORD   <^X0005>        ;*
7052  0006                        .WORD   <^X0006>        ;*
7054
7054  00          ZERO_DAT:       .BYTE   <^X00>          ;*
7055  00          ONE_SHIFT:      .BYTE   <^X00>          ;*
7056  00                          .BYTE   <^X00>          ;*
7057  00                          .BYTE   <^X00>          ;*
7058  01                          .BYTE   <^X01>          ;STANDARD DATA PATTERNS USED
7059  FF          ONE_DAT:        .BYTE   <^XFF>          ; BY VARIOUS TESTS.
```

```
705A  420 FF                            ZERO_SHIFT:     .BYTE     <^XFF>         ;*
705B  421 FF                                            .BYTE     <^XFF>         ;*
705C  422 FF                                            .BYTE     <^XFF>         ;*
705D  423 FE                                            .BYTE     <^XFE>         ;*
705E  424
705E  425 FFFF                          TESTE_DAT:      .WORD     <^XFFFF>       ;*
7060  426 FFFF                                          .WORD     <^XFFFF>       ;*
7062  427 EEEE                                          .WORD     <^XEEEE>       ;*
7064  428 EEEE                                          .WORD     <^XEEEE>       ;*
7066  429 DDDD                                          .WORD     <^XDDDD>       ;*
7068  430 DDDD                                          .WORD     <^XDDDD>       ;*
706A  431 BBBB                                          .WORD     <^XBBBB>       ;*
706C  432 BBBB                                          .WORD     <^XBBBB>       ;*
706E  433 7777                                          .WORD     <^X7777>       ;*
7070  434 7777                                          .WORD     <^X7777>       ;DATA TO BE WRITTEN TO LS
7072  435 0000                                          .WORD     <^X0000>       ; CONSISTS OF ALL ZEROS,
7074  436 0000                                          .WORD     <^X0000>       ; ALL ONES, AND 4 BIT BLOCKS
7076  437 1111                                          .WORD     <^X1111>       ; OF A ONE THROUGH ZEROS
7078  438 1111                                          .WORD     <^X1111>       ; AND A ZERO THROUGH ONES.
707A  439 2222                                          .WORD     <^X2222>       ;*
707C  440 2222                                          .WORD     <^X2222>       ;*
707E  441 4444                                          .WORD     <^X4444>       ;*
7080  442 4444                                          .WORD     <^X4444>       ;*
7082  443 8888                                          .WORD     <^X8888>       ;*
7084  444 8888                                          .WORD     <^X8888>       ;*
7086  445
7086  446
7086  447                               ;*********************************************************************************
7086  448                               ;
7086  449                               ;  SHIFT VARIABLES - DO NOT SEPARATE PAIRS
7086  450                               ;
7086  451                               ;*********************************************************************************
7086  452
7086  453
7086  454 03                            TEMPA_SHIFT:    .BYTE     3                   ;FIRST TEMPORARY STORAGE LOCATION
7087  455 0000008A                      TEMPA_DAT:      .BLKB     3                   ; FOR DATA FROM VARIOUS TESTS.
708A  456
708A  457 04                            TEMPB_SHIFT:    .BYTE     4                   ;SECOND TEMPORARY STORAGE LOCATION
708B  458 0000008F                      TEMPB_DAT:      .BLKB     4                   ; FOR DATA FROM VARIOUS TESTS.
708F  459
708F  460 04                            EXP_SHIFT:      .BYTE     4                   ;LOCATION OF EXPECTED DATA FROM
7090  461 00000094                      EXP_DAT:        .BLKB     4                   ; VARIOUS TESTS.
7094  462
7094  463 04                            REC_SHIFT:      .BYTE     4                   ;LOCATION OF RETURNED DATA FROM
7095  464 00000099                      REC_DAT:        .BLKB     4                   ; VARIOUS TESTS.
7099  465
7099  466
7099  467                               ;*********************************************************************************
7099  468                               ;
7099  469                               ;  CON_SET_FOR_CO  THIS ROUTINE IS USED IF A CONTROL C OCCURS TO SAVE THE
7099  470                               ;                  STATE OF THE MACHINE. IF A CONTINUE IS THEN ISSUED, THIS
7099  471                               ;                  ROUTINE RESTORES THE STATE OF THE MACHINE.
7099  472                               ;
7099  473                               ;*********************************************************************************
7099  474
7099  475
7099  476 0009'3A                       CON_SET_FOR_CO: LDA       LS_ADDR             ;SAVE THE PRESENT LS ADDRESS.
709C  477 000D'32                                       STA       SAVE_LS_ADDR
709F  478
709F  479 0000'CD                                       CALL      SET_FOR_CONT        ;RETURN TO MONITOR.
```

```
70A2  480
70A2  481 000D'3A                                           LDA     SAVE_LS_ADDR    ;RESTORE STATE.
70A5  482 0009'32                                           STA     LS_ADDR
70A8  483
70A8  484 C9                                                RET
70A9  485
70A9  486                              ;*************************************************************************
70A9  487                              ;
70A9  488                              ;  CHECK_RD        THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
70A9  489                              ;                  IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
70A9  490                              ;                  KEEP TRACK OF ELAPSED TIME.
70A9  491                              ;*************************************************************************
70A9  492
70A9  493 40D9 3A                      CHECK_RD:            LDA     MIPFLG                  ;GET STATE OF MODEM IN PROGRESS
70AC  494 B7                                                ORA     A                       ;SET CONDITION CODES
70AD  495 C8                                                RZ                              ;RETURN IF ZERO
70AE  496 0000'CD                                           CALL    GCVECT                  ;ELSE GO TO BOOT CODE IF RD
70B1  497
70B1  498 0000'3A                                           LDA     CNTLC_TYPED             ;SEE IF CONTROL C TYPED (UNDER RD)
70B4  499 0F                                                RRC
70B5  500 0099'DC                                           CC      CON_SET_FOR_CO   ;IF SO, GO TO MONITOR.
70B8  501
70B8  502 C9                                                RET                             ;BACK TO TEST
70B9  503
70B9  504
70B9  505                              ;*************************************************************************
70B9  506                              ;
70B9  507                              ;  LD_SUPPORT  THIS ROUTINE LOADS SUPPORT CODE INTO WCS LOCATIONS 0-13 TO
70B9  508                              ;              ENABLE FAST READS AND WRITES OF WORKING REGISTER 0 AND THUS
70B9  509                              ;              OF LOCAL STORE.
70B9  510                              ;
70B9  511                              ;*************************************************************************
70B9  512
70B9  513
70B9  514 00'32'21                     LD_SUPPORT:          LXI     H,SUPPORT_CODE  ;POINT TO FIRST SUPPORT MICROWORD.
70BC  515 000A'22                                           SHLD    DATA_PNT
70BF  516
70BF  517 00 00 21                                          LXI     H,0             ;POINT TO WCS ADDRESS ZERO.
70C2  518 0000'22                                           SHLD    WCS_ADDRESS
70C5  519
70C5  520 C5 46 00'00'21    ******************               DO      6               ;LOAD FIRST SIX SUPPORT MICROWORDS
          77 06 3E          *
70CD  521                   *                                                       ; FOR WRITING DATA TO WRO.
70CD  522                   *
70CD  523 00A9'CD           *                               CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
70D0  524                   *
70D0  525 000A'2A           *                               LHLD    DATA_PNT        ;MOVE A MICROWORD TO WCS_VALUE.
70D3  526 00'00'11          *                               LXI     D,WCS_VALUE
70D6  527 0000'CD           *                               CALL    MOVER_3
70D9  528                   *
70D9  529 0000'CD           *                               CALL    WCS_WRITE       ;WRITE IT TO WCS.
70DC  530                   *
70DC  531 00'00'21          *                               LXI     H,WCS_ADDRESS   ;POINT TO THE NEXT WCS LOCATION.
70DF  532 0000'CD           *                               CALL    INC_WORD
70E2  533                   *
70E2  534 000A'2A           *                               LHLD    DATA_PNT        ;POINT TO THE NEXT SUPPORT MICROWORD.
70E5  535 23                *                               INX     H
70E6  536 23                *                               INX     H
70E7  537 23                *                               INX     H
70E8  538 000A'22           *                               SHLD    DATA_PNT
```

```
70EB   539                          *
70EB   540 35 00'00'21              ******************          ENDDO
           70 C1 00CD'C2
70F4   541
70F4   542 C5 46 00'00'21           ******************          DO       8                 ;LOAD THE NEXT 8 MICROWORDS FOR
           77 08 3E
70FC   543                          *                                                      ; READING WR[0].
70FC   544                          *
70FC   545 00A9'CD                  *                           CALL     CHECK_RD          ;CALL BOOT CODE IF UNDER RD CONTROL
70FF   546                          *
70FF   547 00'01'21                 *                           LXI      H,X_SHIFT_R+1     ;LOAD A SHIFT RIGHT MICROWORD TO
7102   548 00'00'11                 *                           LXI      D,WCS_VALUE       ; WCS_VALUE.
7105   549 0000'CD                  *                           CALL     MOVER_3
7108   550                          *
7108   551 0000'CD                  *                           CALL     WCS_WRITE         ;WRITE IT TO WCS.
710B   552                          *
710B   553 00'00'21                 *                           LXI      H,WCS_ADDRESS     ;POINT TO THE NEXT WCS LOCATION.
710E   554 0000'CD                  *                           CALL     INC_WORD
7111   555                          *
7111   556 35 00'00'21              ******************          ENDDO
           70 C1 00FC'C2
711A   557
711A   558 C9                                                   RET
711B   559
711B   560
711B   561
711B   562                                         ;*************************************************************************
711B   563                                         ;
711B   564                                         ;  SET_COMPARE   THIS ROUTINE READS THE UPC AND PUTS IT IN REC_DAT+2 AND READS
711B   565                                         ;                THE CORRESPONDING VERIFY WRITER TABLE ENTRY AND PUTS IT IN
711B   566                                         ;                EXP_DAT+2. THE TABLE POINTER IS INCREMENTED.
711B   567                                         ;
711B   568                                         ;*************************************************************************
711B   569
711B   570
711B   571 0000'CD                                SET_COMPARE:     CALL     LOAD_UPC          ;PUT THE UPC CONTENTS IN REC_DAT.
711E   572 0000'2A                                                 LHLD     UPC_VALUE
7121   573 0097'22                                                 SHLD     REC_DAT+2
7124   574 0000'CD                                                 CALL     LOAD_UPC          ;AND RELOAD THE UPC.
7127   575
7127   576 0000'2A                                                 LHLD     WRITE_TAB_PNT     ;LOAD AN ENTRY FROM THE VERIFY WRITE
712A   577 7E                                                      MOV      A,M               ; WRITE TABLE INTO EXP_DAT.
712B   578 0093'32                                                 STA      EXP_DAT+3
712E   579 23                                                      INX      H
712F   580 7E                                                      MOV      A,M
7130   581 0092'32                                                 STA      EXP_DAT+2
7133   582
7133   583 23                                                      INX      H                 ;INCREMENT THE VERIFY WRITE TABLE
7134   584 0000'22                                                 SHLD     WRITE_TAB_PNT     ; POINTER.
7137   585
7137   586 C9                                                      RET
7138   587
7138   588
7138   589                                         ;*************************************************************************
7138   590                                         ;
7138   591                                         ;  LS_WRITE   THIS ROUTINE WRITES FOUR BYTES OF DATA FROM DATA0-DATA3 TO LOCAL
7138   592                                         ;             STORE AT THE LOCATION GIVEN BY THE CONTENTS OF THE ACCUMUALTOR.
7138   593                                         ;
7138   594                                         ;*************************************************************************
7138   595
```

```
7138
7138  0000'32          LS_WRITE:     STA     XD_ADDRS          ;STORE THE LS ADDRESS IN XD_ADDRS.
713B
713B  80 E6                          ANI     <^X80>            ;SAVE THE MSB.
713D  07                             RLC                       ;ROTATE LEFT ONE PLACE.
713E  BE F6                          ORI     <^XBE>            ;OR IN THE MOV WR TO LS OP CODE.
7140  0017'32                        STA     MOV_WRO_LSX       ;STORE BIT 7 OF EXTENDED ADDRESS IN
7143                                                           ; BIT 16 OF INSTRUCTION.
7143
7143  0000'3A                        LDA     XD_ADDRS          ;LOAD EXTENDED ADDRESS.
7146  B7                             ORA     A                 ;CLEAR THE CARRY.
7147  17                             RAL                       ;SHIFT LEFT TO CLEAR LSB (B FIELD).
7148  0018'32                        STA     MOV_WRO_LSX+1     ;STORE BITS 0-6 OF EXTENDED ADDRESS
714B                                                           ; IN BITS 9-15 OF INSTRUCTION.
714B
714B  0172'CD                        CALL    FAST_WRITE_WRO    ;WRITE DATA TO WORKING REG 0.
714E
714E  00'17'21                       LXI     H,MOV_WRO_LSX     ;MOVE THE DATA FROM WRO TO LS.
7151  0294'CD                        CALL    EXEC_INST
7154
7154  C9                             RET
7155
7155
7155                   ;************************************************************************
7155                   ;
7155                   ;  LS_READ  THIS ROUTINE READS FOUR BYTES OF DATA FROM THE LS LOCATION
7155                   ;           POINTED TO BY THE ACCUMULATOR TO DATA0-DATA3.
7155                   ;
7155                   ;************************************************************************
7155
7155
7155  0000'32          LS_READ:      STA     XD_ADDRS          ;STORE THE LS ADDRESS IN XD_ADDRS.
7158
7158  80 E6                          ANI     <^X80>            ;SAVE THE MSB.
715A  07                             RLC                       ;ROTATE LEFT ONE PLACE.
715B  36 F6                          ORI     <^X36>            ;OR IN THE MOV WRO TO LS OP CODE.
715D  001A'32                        STA     MOV_LSX_WRO       ;STORE BIT 7 OF EXTENDED ADDRESS IN
7160                                                           ; BIT 16 OF INSTRUCTION.
7160
7160  0000'3A                        LDA     XD_ADDRS          ;LOAD EXTENDED ADDRESS.
7163  B7                             ORA     A                 ;CLEAR THE CARRY.
7164  17                             RAL                       ;SHIFT LEFT TO CLEAR LSB (B FIELD).
7165  001B'32                        STA     MOV_LSX_WRO+1     ;STORE BITS 0-6 OF EXTENDED ADDRESS
7168                                                           ; IN BITS 9-15 OF INSTRUCTION.
7168
7168  00'1A'21                       LXI     H,MOV_LSX_WRO     ;EXECUTE THE INSTRUCTION.
716B  0294'CD                        CALL    EXEC_INST
716E
716E  01B9'CD                        CALL    FAST_READ_WRO     ;READ WRO.
7171
7171  C9                             RET
7172
7172
7172                   ;************************************************************************
7172                   ;
7172                   ;  FAST_WRITE_WRO  THIS ROUTINE PERFORMS A FAST WRITE OF WORKING REGISTER
7172                   ;                  ZERO BY USING SUPPORT MICROCODE IN WCS.
7172                   ;
7172                   ;************************************************************************
7172
```

```
7172  656
7172  657 00 00 21                                   FAST_WRITE_WRO: LXI     H,0             ;PREPARE TO EXECUTE WCS SUPPORT CODE
7175  658 0000'22                                                    SHLD    UPC_VALUE       ; STORED IN WCS LOACATIONS 0-5.
7178  659 0000'CD                                                    CALL    LOAD_UPC
717B  660
717B  661 0000'CD                                                    CALL    NOP_CSR         ;LOAD CSR WITH NOP INSTRUCTION.
717E  662
717E  663 23 D3                                                      OUT     SETSS           ;EXECUTE WCS INSTRUCTION TO CLEAR WRO.
7180  664 22 D3                                                      OUT     CLRSS
7182  665
7182  666 23 D3                                                      OUT     SETSS           ;COMPLEMENT WRO.
7184  667 22 D3                                                      OUT     CLRSS
7186  668
7186  669 00A9'CD                                                    CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7189  670
7189  671 C5 46 00'00'21      ******************                     DO      32
          77 20 3E            *
7191  672                     *
7191  673 00'04'21            *                                      LXI     H,DATA_SHIFT+4  ;POINT TO END OF DATA AREA.
7194  674 04 0E               *                                      MVI     C,4             ;SHIFT FOUR BYTES INTO WRO.
7196  675 0000'CD            *                                      CALL    SHIFTER1
7199  676                     *
7199  677 AB D3               *                                      OUT     CLRATN          ;SET A ZERO TO BE READ.
719B  678                     *
719B  679 01A0'D2            * ****************                     IFC
719E  680 AA D3               * *                                    OUT     SETATN          ;SET A ONE TO BE READ.
71A0  681                     * ****************                     ENDIF
71A0  682                     *
71A0  683 23 D3               *                                      OUT     SETSS           ;EXECUTE WCS SUBROUTINE TO SHIFT
71A2  684 22 D3               *                                      OUT     CLRSS           ; A BIT INTO WRO.
71A4  685                     *
71A4  686 23 D3               *                                      OUT     SETSS           ;*
71A6  687 22 D3               *                                      OUT     CLRSS
71A8  688                     *
71A8  689 23 D3               *                                      OUT     SETSS           ;*
71AA  690 22 D3               *                                      OUT     CLRSS
71AC  691                     *
71AC  692 35 00'00'21         ******************                     ENDDO
          70 C1 0191'C2
71B5  693
71B5  694 00A9'CD                                                    CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
71B8  695
71B8  696 C9                                                         RET
71B9  697
71B9  698
71B9  699                                            ;*****************************************************************************
71B9  700                                            ;
71B9  701                                            ;  FAST_READ_WRO  THIS ROUTINE PERFORMS A FAST READ OF WORKING REGISTER ZERO
71B9  702                                            ;                 USING SUPPORT MICROCODE IN WCS.
71B9  703                                            ;
71B9  704                                            ;*****************************************************************************
71B9  705
71B9  706
71B9  707 00'00'21                                   FAST_READ_WRO:  LXI     H,X_MOV_WRWR    ;ENABLE WRO TO BE READ.
71BC  708 029C'CD                                                    CALL    LD_CSR_INST
71BF  709
71BF  710 EC D3                                                      OUT     YBUSRD          ;LATCH THE DATA IN THE Y-BUS LATCH.
71C1  711 80 DB                                                      IN      READ            ;READ THE Y-BUS.
71C3  712 0000'32                                                    STA     DATA3           ;STORE IN DATA3.
71C6  713
```

```
71C6  01D9'CD                                      CALL    READ_BYTE_WR0   ;READ THE NEXT BYTE OF WR0.
71C9  0000'32                                      STA     DATA2           ;STORE IN DATA2.
71CC
71CC  01D9'CD                                      CALL    READ_BYTE_WR0   ;AND THE NEXT...
71CF  0000'32                                      STA     DATA1
71D2
71D2  01D9'CD                                      CALL    READ_BYTE_WR0   ;AND THE LAST.
71D5  0000'32                                      STA     DATA0
71D8
71D8  C9                                           RET
71D9
71D9
71D9                                   ;*****************************************************************************
71D9                                   ;
71D9                                   ;  READ_BYTE_WR0  THIS ROUTINE EXECUTES SUPPORT CODE OUT OF WCS WHICH SHIFTS
71D9                                   ;                 THE CONTENTS OF WORKING REGISTER ZERO TO THE RIGHT EIGHT
71D9                                   ;                 PLACES. IT THEN READS THE LOW BYTE OF WR0 AND LEAVES IT
71D9                                   ;                 IN THE ACCUMULATOR.
71D9                                   ;
71D9                                   ;
71D9                                   ;*****************************************************************************
71D9
71D9
71D9  06 00 21                         READ_BYTE_WR0:  LXI     H,<^X0600>      ;PREPARE TO EXECUTE WCS SUPPORT CODE
71DC  0000'22                                          SHLD    UPC_VALUE       ; STORED IN WCS LOACATIONS 6-D.
71DF  0000'CD                                          CALL    LOAD_UPC
71E2
71E2  0000'CD                                          CALL    NOP_CSR         ;LOAD CSR WITH NOP INSTRUCTION.
71E5
71E5  C5 46 00'00'21       ******************          DO      9
      77 09 3E             *
71ED                       *
71ED  23 D3                *                           OUT     SETSS           ;EXECUTE A SHIFT INSTRUCTION.
71EF  22 D3                *                           OUT     CLRSS
71F1                       *
71F1  35 00'00'21          ******************          ENDDO
      70 C1 01ED'C2
71FA
71FA  00'00'21                                         LXI     H,X_MOV_WRWR    ;ENABLE WR0 TO BE READ.
71FD  029C'CD                                          CALL    LD_CSR_INST
7200
7200  EC D3                                            OUT     YBUSRD          ;LATCH THE DATA IN THE Y-BUS LATCH.
7202  80 DB                                            IN      READ            ;READ THE Y-BUS.
7204
7204  C9                                               RET
7205
7205
7205                                   ;*****************************************************************************
7205                                   ;
7205                                   ;  WRITE_WR1_32  THIS ROUTINE LOADS 4 BYTES OF DATA FROM DATA0 - DATA3 TO THE
7205                                   ;                2901'S WR[1].
7205                                   ;
7205                                   ;*****************************************************************************
7205
7205
7205  00'29'21                         WRITE_WR1_32:   LXI     H,X_CLR_WR1     ;CLEAR WR1.
7208  0294'CD                                          CALL    EXEC_INST
720B
720B  00'2C'21                                         LXI     H,X_COM_WR1     ;COMPLEMENT WR1.
720E  0294'CD                                          CALL    EXEC_INST
7211
```

```
7211  772 C5 46 00'00'21    ******************                DO      32
          77 20 3E
7219  773                   *
7219  774 00'00'21          *                                 LXI     H,DATA_SHIFT    ;PREPARE TO SHIFT DATA INTO WR1.
721C  775 0000'22           *                                 SHLD    SHIFT_PNT
721F  776                   *
721F  777 0000'CD           *                                 CALL    SHIFTER         ;SHIFT A BIT OF DATA INTO THE CARRY.
7222  778                   *
7222  779 022E'D2           * ****************                IFC                     ;IF THE CARRY IS SET...
7225  780                   * *
7225  781 00'23'21          * *                               LXI     H,ROT_WR1_L     ;...SHIFT A ONE INTO WR1.
7228  782 0294'CD           * *                               CALL    EXEC_INST
722B  783                   * *
722B  784 0234'C3           * ****************                ELSE                    ;OTHERWISE...
722E  785                   * *
722E  786 00'26'21          * *                               LXI     H,SHIFT_WR1_L   ;...SHIFT A ZERO INTO WR1.
7231  787 0294'CD           * *                               CALL    EXEC_INST
7234  788                   * *
7234  789                   * ****************                ENDIF
7234  790                   *
7234  791 35 00'00'21       ******************                ENDDO
          70 C1 0219'C2
723D  792
723D  793 C9                                                  RET
723E  794
723E  795
723E  796                                       :****************************************************************************
723E  797                                       :
723E  798                                       :  READ_WR1_32  THIS ROUTINE IS USED TO READ THE CONTENTS OF THE 2901'S
723E  799                                       :               WR[1]. THE DATA IS PUT INTO LOCATIONS DATA0 - DATA3.
723E  800                                       :
723E  801                                       :****************************************************************************
723E  802
723E  803
723E  804 00'1D'21                              READ_WR1_32:    LXI     H,MOV_WR1_WR1   ;LOAD THE CSR WITH AN INSTRUCTION
7241  805 029C'CD                                               CALL    LD_CSR_INST     ; TO ENABLE WR1 TO BE READ.
7244  806
7244  807 EC D3                                                 OUT     YBUSRD          ;READ THE LOW BYTE AND SHIFT
7246  808 80 DB                                                 IN      READ            ; THE CONTENTS OF WR1 TO THE
7248  809 0000'32                                               STA     DATA3           ; RIGHT EIGHT BITS.
724B  810 027C'CD                                               CALL    SHIFT_WR1_R8
724E  811
724E  812 00'1D'21                                              LXI     H,MOV_WR1_WR1   ;REPEAT FOR NEXT BYTE.
7251  813 029C'CD                                               CALL    LD_CSR_INST
7254  814
7254  815 EC D3                                                 OUT     YBUSRD
7256  816 80 DB                                                 IN      READ
7258  817 0000'32                                               STA     DATA2
725B  818 027C'CD                                               CALL    SHIFT_WR1_R8
725E  819
725E  820 00'1D'21                                              LXI     H,MOV_WR1_WR1   ;REPEAT FOR NEXT BYTE.
7261  821 029C'CD                                               CALL    LD_CSR_INST
7264  822
7264  823 EC D3                                                 OUT     YBUSRD
7266  824 80 DB                                                 IN      READ
7268  825 0000'32                                               STA     DATA1
726B  826 027C'CD                                               CALL    SHIFT_WR1_R8
726E  827
726E  828 00'1D'21                                              LXI     H,MOV_WR1_WR1   ;REPEAT FOR LAST BYTE.
7271  829 029C'CD                                               CALL    LD_CSR_INST
```

```
7274  830
7274  831 EC D3                                               OUT     YBUSRD
7276  832 80 DB                                               IN      READ
7278  833 0000'32                                             STA     DATA0
727B  834
727B  835 C9                                                  RET
727C  836
727C  837
727C  838                                     ;*************************************************************************
727C  839                                     :
727C  840                                     :  SHIFT_WR1_R8:  THIS ROUTINE SHIFTS THE CONTENTS OF THE 2901'S WR[1] RIGHT
727C  841                                     :                 EIGHT BITS.
727C  842                                     :
727C  843                                     ;*************************************************************************
727C  844
727C  845
727C  846 C5 46 00'00'21     ***************** SHIFT_WR1_R8:   DO      8
          77 08 3E           *
7284  847                    *
7284  848 00'20'21           *                                LXI     H,SHIFT_WR1_R   ;SHIFT WR1 RIGHT ONE BIT.
7287  849 0294'CD            *                                CALL    EXEC_INST
728A  850                    *
728A  851 35 00'00'21        *****************                ENDDO
          70 C1 0284'C2
7293  852
7293  853 C9                                                  RET
7294  854
7294  855
7294  856                                     ;*************************************************************************
7294  857                                     :
7294  858                                     :  EXEC_INST  THIS ROUTINE EXECUTES AN INSTRUCTION POINTED TO BY THE H,L
7294  859                                     :             PAIR.
7294  860                                     :
7294  861                                     ;*************************************************************************
7294  862
7294  863
7294  864 029C'CD                             EXEC_INST:      CALL    LD_CSR_INST     ;LOAD INSTRUCTION INTO THE CSR.
7297  865
7297  866 23 D3                                               OUT     SETSS           ;CLOCK THE CPU.
7299  867 22 D3                                               OUT     CLRSS
729B  868
729B  869 C9                                                  RET
729C  870
729C  871
729C  872                                     ;*************************************************************************
729C  873                                     :
729C  874                                     :  LD_CSR_INST  THIS ROUTINE LOADS THE CSR WITH AN INSTRUCTION, WHICH IS
729C  875                                     :               POINTED TO BY THE CONTENTS OF THE H,L PAIR.
729C  876                                     :
729C  877                                     ;*************************************************************************
729C  878
729C  879
729C  880 00'00'11                            LD_CSR_INST:    LXI     D,CSR_VALUE     ;MOVE INSTRUCTION POINTED TO BY THE
729F  881 0000'CD                                             CALL    MOVER_3         ; H,L PAIR INTO CSR_VALUE.
72A2  882
72A2  883 A2 D3                                               OUT     VSHIFT          ;SET V-BUS TO SHIFTING MODE.
72A4  884
72A4  885 00'00'21                                            LXI     H,CSR_SHIFT     ;PREPARE TO SHIFT DATA INTO CSR.
72A7  886 0000'22                                             SHLD    SHIFT_PNT
72AA  887
```

```
72AA  888 C5 46 00'00'21      ********************                DO      24
          77 18 3E
72B2  889                     *
72B2  890 38 D3               *                                   OUT     CLRCSR          ;CLEAR THE CSR SERIAL INPUT.
72B4  891                     *
72B4  892 87 DB               *                                   IN      CSR23           ;READ THE CSR'S MSB.
72B6  893 1F                  *                                   RAR                     ;ROTATE IT INTO THE CARRY.
72B7  894 0000'CD             *                                   CALL    SHIFTER         ;SHIFT THE DATA THROUGH CSR_VALUE.
72BA  895                     *
72BA  896 02BF'D2             *                                   JNC     1$              ;IF THE CARRY IS SET...
72BD  897                     *
72BD  898 39 D3               *                                   OUT     SETCSR          ;SET THE CSR SERIAL INPUT.
72BF  899                     *
72BF  900 25 D3               *                     1$:           OUT     SETCSK          ;CLOCK THE CSR SERIAL LOAD.
72C1  901 24 D3               *                                   OUT     CLRCSK
72C3  902                     *
72C3  903 35 00'00'21         ********************                ENDDO
          70 C1 02B2'C2
72CC  904
72CC  905 A3 D3                                                   OUT     VNORML          ;SET V-BUS TO NORMAL MODE.
72CE  906
72CE  907 00A9'CD                                                 CALL    CHECK_RD        ;CALL BOOT CODE IF RD IN CONTROL
72D1  908
72D1  909 C9                                                      RET
72D2  910
72D2  911
72D2  912                                           ;******************************************************************************
72D2  913                                           ;
72D2  914                                           ;  LOAD_MOD_DAT THIS ROUTINE LOADS THE MODULE DATA POINTED TO BY THE H,L
72D2  915                                           ;               PAIR INTO CON_MOD_DAT.
72D2  916                                           ;
72D2  917                                           ;******************************************************************************
72D2  918
72D2  919
72D2  920 00'00'11                                  LOAD_MOD_DAT: LXI     D,CON_MOD_DAT   ;MOVE THE MODULE DATA POINTED TO BY
72D5  921 05 0E                                                   MVI     C,5             ; THE H,L PAIR TO CON_MOD_DAT.
72D7  922 0000'CD                                                 CALL    MOVER
72DA  923
72DA  924 C9                                                      RET
72DB  925
72DB  926
72DB  927                                           ;******************************************************************************
72DB  928                                           ;
72DB  929                                           ;  TEST9_1_0    THIS ROUTINE IS USED BY TEST 9 TO WRITE AN 8-BIT DATA PATTERN
72DB  930                                           ;               INTO THE ALU AND THEN READ IT BACK
72DB  931                                           ;
72DB  932                                           ;******************************************************************************
72DB  933
72DB  934 0090'32                                   TEST9_1_0:    STA     EXP_DAT         ;LOAD DATA PATTERN INTO EXP_DAT.
72DE  935 CC D3                                                   OUT     WRITE           ;WRITE THIS PATTERN TO THE D-BUS.
72E0  936
72E0  937 23 D3                                                   OUT     SETSS           ;CLOCK THE CPU TO ENABLE WR OUTPUT.
72E2  938 22 D3                                                   OUT     CLRSS
72E4  939
72E4  940 00'00'21                                                LXI     H,X_MOV_WRWR    ;LOAD THE CSR WITH AN INSTRUCTION TO
72E7  941 029C'CD                                                 CALL    LD_CSR_INST     ; ENABLE THE WR TO BE READ.
72EA  942
72EA  943 EC D3                                                   OUT     YBUSRD          ;CLOCK THE Y-BUS LATCH.
72EC  944 80 DB                                                   IN      READ            ;READ THE Y-BUS CONTENTS.
72EE  945 0095'32                                                 STA     REC_DAT         ;STORE THE Y-BUS CONTENTS IN REC_DAT.
```

```
72F1  946
72F1  947 00'90'21            ******************          IFNEQ   EXP_DAT,REC_DAT,1   ;COMPARE THE EXPECTED AND RECEIVED
          01 0E 00'95'11      *
          0302'CA 0000'CD     *
72FF  948                     *                                                   ; DATA.
72FF  949                     *
72FF  950 037B'CD             *                           CALL    TEST9_ERROR     ;PRINT AN ERROR MESSAGE.
7302  951                     *
7302  952                     ******************          ENDIF
7302  953
7302  954 0000'3A                                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7305  955 0F                                              RRC                     ;RETURN WITH C SET IF LOOP ON ERROR
7306  956 C9                                              RET                     ; AND C CLEAR IF NO LOOP
7307  957
7307  958
7307  959
7307  960                                     ;*************************************************************************
7307  961                                     ;
7307  962                                     ;  TEST9_SHIFT  THIS ROUTINE IS USED BY TEST 9 TO WRITE AN 8-BIT SHIFT PATTERN
7307  963                                     ;               INTO THE ALU AND THEN READ IT BACK
7307  964                                     ;
7307  965                                     ;*************************************************************************
7307  966
7307  967 00'0E'21                            TEST9_SHIFT:   LXI     H,MOV_CCR_WRO   ;LOAD THE CSR WITH A MOVE INSTRUCTION
730A  968 029C'CD                                            CALL    LD_CSR_INST     ; USED TO PASS DATA THROUGH THE ALU.
730D  969
730D  970 0087'3A                                            LDA     TEMPA_DAT       ;LOAD THE DATA INTO A AND INTO
7310  971 0090'32                                            STA     EXP_DAT         ; EXP_DAT.
7313  972 CC D3                                              OUT     WRITE           ;WRITE THIS PATTERN TO THE D-BUS.
7315  973
7315  974 23 D3                                              OUT     SETSS           ;CLOCK THE CPU TO ENABLE WR OUTPUT.
7317  975 22 D3                                              OUT     CLRSS
7319  976
7319  977 00'00'21                                           LXI     H,X_MOV_WRWR    ;LOAD THE CSR WITH AN INSTRUCTION TO
731C  978 029C'CD                                            CALL    LD_CSR_INST     ; ENABLE THE WR TO BE READ.
731F  979
731F  980 EC D3                                              OUT     YBUSRD          ;CLOCK THE Y-BUS LATCH.
7321  981 80 DB                                              IN      READ            ;READ THE Y-BUS CONTENTS.
7323  982 0095'32                                            STA     REC_DAT         ;STORE THE Y-BUS CONTENTS IN REC_DAT.
7326  983
7326  984 00'90'21            ******************          IFNEQ   EXP_DAT,REC_DAT,1   ;COMPARE THE EXPECTED AND RECEIVED
          01 0E 00'95'11      *
          0337'CA 0000'CD     *
7334  985                     *                                                   ; DATA.
7334  986                     *
7334  987 037B'CD             *                           CALL    TEST9_ERROR     ;PRINT AN ERROR MESSAGE.
7337  988                     *
7337  989                     ******************          ENDIF
7337  990
7337  991 0000'3A                                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
733A  992 0F                                              RRC
733B  993 0307'DA                                         JC      TEST9_SHIFT     ;...JUMP TO BEGINNING OF ERROR LOOP.
733E  994 C9                                              RET
733F  995
733F  996
733F  997
733F  998                                     ;*************************************************************************
733F  999                                     ;
733F 1000                                     ;  TEST9_2_0    THIS ROUTINE IS USED BY TEST 9 TO WRITE AN 8 BIT DATA PATTERN
733F 1001                                     ;               INTO THE ALU AND THEN READ IT BACK WHILE TESTING WR CRR
```

```
733F 1002                                               ;
733F 1003                                               ;*************************************************************************
733F 1004
733F 1005 0090'32                                       TEST9_2_0:      STA     EXP_DAT         ;LOAD PATTERN INTO EXP_DAT
7342 1006
7342 1007 00'0E'21                                                      LXI     H,MOV CCR WR0   ;LOAD THE CSR WITH A MOVE INSTRUCTION
7345 1008 029C'CD                                                       CALL    LD_CSR_INST     ; USED TO PASS DATA THROUGH THE ALU
7348 1009
7348 1010 0090'3A                                                       LDA     EXP_DAT
734B 1011
734B 1012 CC D3                                                         OUT     WRITE           ;WRITE PATTERN TO THE D-BUS
734D 1013
734D 1014 23 D3                                                         OUT     SETSS           ;CLOCK DATA PATH
734F 1015 22 D3                                                         OUT     CLRSS
7351 1016
7351 1017 00'2F'21                                                      LXI     H,MISC2 WR CRR  ;LOAD THE CSR WITH A MISC INSTRUCTION
7354 1018 029C'CD                                                       CALL    LD_CSR_INST     ; USED TO ENABLE WR CRR
7357 1019
7357 1020 23 D3                                                         OUT     SETSS           ;CLOCK DATA PATH
7359 1021 22 D3                                                         OUT     CLRSS
735B 1022
735B 1023 80 DB                                                         IN      READ            ;READ THE Y-BUS CONTENTS
735D 1024
735D 1025 0095'32                                                       STA     REC_DAT         ;STORE THE Y-BUS CONTENTS IN REC_DAT
7360 1026
7360 1027 00'90'21              ******************                      IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND RECEIVED
          01 0E 00'95'11        *
          0376'CA 0000'CD       *
736E 1028                       *                                                                       ; DATA
736E 1029                       *
736E 1030 02 3E                 *                                       MVI     A,2             ;ERROR NUMBER 2
7370 1031 0000'32               *                                       STA     CON_ERR_NUM
7373 1032                       *
7373 1033 0380'CD               *                                       CALL    TEST9_ERR2_3    ;REPORT ERROR
7376 1034                       *
7376 1035                       ******************                      ENDIF
7376 1036
7376 1037 0000'3A                                                       LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7379 1038 0F                                                            RRC                     ;RETURN WITH C SET IF LOOP ON ERROR
737A 1039 C9                                                            RET                     ; AND C CLEAR IF NO LOOP
737B 1040
737B 1041
737B 1042
737B 1043                                               ;*************************************************************************
737B 1044                                               ;
737B 1045                                               ;  TEST9_ERROR  THIS ROUTINE IS USED BY TEST 9 TO DEFINE THE ERROR VARIABLES
737B 1046                                               ;               IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
737B 1047                                               ;
737B 1048                                               ;
737B 1049                                               ;*************************************************************************
737B 1050
737B 1051 01 3E                                         TEST9_ERROR:    MVI     A,1             ;ERROR NUMBER 1.
737D 1052 0000'32                                                       STA     CON_ERR_NUM
7380 1053
7380 1054 0090'3A                                       TEST9_ERR2_3:   LDA     EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
7383 1055 0003'32                                                       STA     CON_EXP_DAT+3
7386 1056
7386 1057 0095'3A                                                       LDA     REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
7389 1058 0003'32                                                       STA     CON_REC_DAT+3
738C 1059
```

```
738C 1060 FF FF 21                                      LXI     H,<^XFFFF>        ;ONLY THE LAST BYTE IS OF
738F 1061 0000'22                                       SHLD    CON_MSK_DAT       ; INTEREST.
7392 1062 00 FF 21                                      LXI     H,<^X00FF>
7395 1063 0002'22                                       SHLD    CON_MSK_DAT+2
7398 1064
7398 1065 0000'32 3C AF                                 SET     NA_OTHER          ;NO ADDITIONAL DATA.
739D 1066
739D 1067 0446'C3                                       JMP     REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD
73A0 1068
73A0 1069
73A0 1070                               ;*****************************************************************************
73A0 1071                               ;
73A0 1072                               ;  TESTA_ERROR  THIS ROUTINE IS USED BY TEST A TO DEFINE THE ERROR VARIABLES
73A0 1073                               ;               IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
73A0 1074                               ;
73A0 1075                               ;*****************************************************************************
73A0 1076
73A0 1077
73A0 1078 0090'3A                      TESTA_ERROR:    LDA     EXP_DAT           ;MOVE EXPECTED DATA TO CON_EXP_DAT.
73A3 1079 0003'32                                       STA     CON_EXP_DAT+3
73A6 1080
73A6 1081 0095'3A                                       LDA     REC_DAT           ;MOVE RECEIVED DATA TO CON_REC_DAT.
73A9 1082 0003'32                                       STA     CON_REC_DAT+3
73AC 1083
73AC 1084 FF FF 21                                      LXI     H,<^XFFFF>        ;ERROR MASK INCLUDES ONLY THE LEAST
73AF 1085 0000'22                                       SHLD    CON_MSK_DAT       ; SIGNIFICANT BYTE.
73B2 1086 00 FF 21                                      LXI     H,<^X00FF>
73B5 1087 0002'22                                       SHLD    CON_MSK_DAT+2
73B8 1088
73B8 1089 03 0E 00'00'21                                ZERO    CON_ADD_DAT,3     ;ADDITIONAL DATA CONTAINS THE SHIFT
          0D 23 77 AF
          03BE'C2
73C4 1090 0008'3A                                       LDA     SHIFT_CNT_DAT     ; DEPTH.
73C7 1091 0003'32                                       STA     CON_ADD_DAT+3
73CA 1092
73CA 1093 0446'C3                                       JMP     REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD
73CD 1094
73CD 1095
73CD 1096                               ;*****************************************************************************
73CD 1097                               ;
73CD 1098                               ;  TESTB_ERROR  THIS ROUTINE IS USED BY TEST B TO DEFINE THE ERROR VARIABLES
73CD 1099                               ;               IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
73CD 1100                               ;
73CD 1101                               ;*****************************************************************************
73CD 1102
73CD 1103
73CD 1104 00'90'21                      TESTB_ERROR:    LXI     H,EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
73D0 1105 00'00'11                                      LXI     D,CON_EXP_DAT
73D3 1106 0000'CD                                       CALL    MOVER_4
73D6 1107
73D6 1108 00'95'21                                      LXI     H,REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
73D9 1109 00'00'11                                      LXI     D,CON_REC_DAT
73DC 1110 0000'CD                                       CALL    MOVER_4
73DF 1111
73DF 1112 0446'C3                                       JMP     REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD
73E2 1113
73E2 1114
73E2 1115                               ;*****************************************************************************
73E2 1116                               ;
73E2 1117                               ;  TESTD_ERROR  THIS ROUTINE IS USED BY TEST D TO DEFINE THE ERROR VARIABLES
```

```
73E2 1118                               ;               IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
73E2 1119                               ;
73E2 1120                               ;  TESTC_ERROR  SAME FOR TEST C
73E2 1121                               ;
73E2 1122                               ;****************************************************************************
73E2 1123
73E2 1124
73E2 1125 01 3E                          TESTD_ERROR:    MVI     A,1             ;ERROR NUMBER 1.
73E4 1126 0000'32                                        STA     CON_ERR_NUM
73E7 1127
73E7 1128 00'90'21                       TESTC_ERROR:    LXI     H,EXP_DAT       ;MOVE EXPECTED DATA TO CON_EXP_DAT.
73EA 1129 00'00'11                                      LXI     D,CON_EXP_DAT
73ED 1130 0000'CD                                       CALL    MOVER_4
73F0 1131
73F0 1132 00'95'21                                      LXI     H,REC_DAT       ;MOVE RECEIVED DATA TO CON_REC_DAT.
73F3 1133 00'00'11                                      LXI     D,CON_REC_DAT
73F6 1134 0000'CD                                       CALL    MOVER_4
73F9 1135
73F9 1136 0000'32 3C AF                                 SET     NA_OTHER        ;NO ADDITIONAL DATA.
73FE 1137
73FE 1138 04 0E 00'00'21                                ZERO    CON_MSK_DAT,4   ;ERROR MASK INCLUDES ALL BITS.
          0D 23 77 AF
          0404'C2
740A 1139
740A 1140 0446'C3                                       JMP     REPORT_CPU_ERROR ;REPORT ERROR ON CPU BOARD
740D 1141
740D 1142
740D 1143                               ;****************************************************************************
740D 1144                               ;
740D 1145                               ;  TESTE_F_ERROR  THIS ROUTINE IS USED BY TESTS E AND F TO DEFINE THE ERROR
740D 1146                               ;                 VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
740D 1147                               ;
740D 1148                               ;****************************************************************************
740D 1149
740D 1150
740D 1151 00'90'21                       TESTE_F_ERROR:  LXI     H,EXP_DAT       ;MOVE EXPECTED DATA TO CON_EXP_DAT.
7410 1152 00'00'11                                      LXI     D,CON_EXP_DAT
7413 1153 0000'CD                                       CALL    MOVER_4
7416 1154
7416 1155 00'00'21                                      LXI     H,DATA0         ;MOVE RECEIVED DATA TO CON_REC_DAT.
7419 1156 00'00'11                                      LXI     D,CON_REC_DAT
741C 1157 0000'CD                                       CALL    MOVER_4
741F 1158
741F 1159 03 0E 00'00'21                                ZERO    CON_ADD_DAT,3   ;ADDITIONAL DATA CONTAINS LS
          0D 23 77 AF
          0425'C2
742B 1160 0009'3A                                       LDA     LS_ADDR         ; POINTER.
742E 1161 0003'32                                       STA     CON_ADD_DAT+3
7431 1162
7431 1163 04 0E 00'00'21                                ZERO    CON_MSK_DAT,4   ;ERROR MASK INCLUDES ALL BITS.
          0D 23 77 AF
          0437'C2
743D 1164
743D 1165 00'00'21                                      LXI     H,MWCS_A        ;WCS MODULE BEING TESTED.
7440 1166 02D2'CD                                       CALL    LOAD_MOD_DAT
7443 1167
7443 1168 044C'C3                                       JMP     REPORT_ERROR    ;REPORT ERROR
7446 1169
7446 1170                               ;****************************************************************************
7446 1171                               ;
```

```
7446                    ;  REPORT_CPU_ERROR  THIS ROUTINE LOADS MODULE DATA WITH CODE FOR CPU
7446                    ;                    BOARD AND CALLS THE ERROR REPORT ROUTINE
7446                    ;
7446                    ;  REPORT_ERROR      THIS ROUTINE CALLS THE ERROR REPORT ROUTINE ONLY
7446                    ;
7446                    ;*******************************************************************************
7446
7446                    REPORT_CPU_ERROR:
7446  00'00'21                          LXI     H,MCPU_A          ;CPU MODULE BEING TESTED.
7449  02D2'CD                           CALL    LOAD_MOD_DAT
744C
744C  0000'CD           REPORT_ERROR:   CALL    CON_ERROR         ;PRINT AN ERROR.
744F
744F  C9                                RET
7450
7450
7450
7450                    ;*******************************************************************************
7450                    ;
7450                    ;
7450                    ;
7450                    ;       TEST9       2901 DATA PATH TEST (WORKING REG 0 AND WR CRR TESTS)
7450                    ;       -----       ----------------------------------------------------
7450                    ;
7450                    ;               DATA WILL BE SENT TO A 2901 WORKING REGISTER OVER THE
7450                    ;       8 BIT DATA PATH AND THEN READ BACK.  THE DATA SENT WILL BE ALL 1'S,
7450                    ;       ALL 0'S, ONE RIPPLED THROUGH A FIELD OF ZEROS AND A ZERO RIPPLED
7450                    ;       THROUGH A FIELD OF ONES.
7450                    ;
7450                    ;
7450                    ;       TEST STEPS:
7450                    ;
7450                    ;               1) WRITE DATA TO WORKING REG.
7450                    ;               2) READ DATA BACK.
7450                    ;               3) PRINT ERRORS IF ANY.
7450                    ;               4) REPEAT 1-3 FOR ALL PATTERNS.
7450                    ;               5) REPEAT, CLOCKING LATCH WITH WR CRR
7450                    ;               6) TEST FOR CLOCK LOGIC STUCK SET
7450                    ;
7450                    ;
7450                    ;       ASSUMPTIONS: THE D AND Y-BUSES ARE WORKING AS TESTED IN THE
7450                    ;                    PREVIOUS TEST.
7450                    ;
7450                    ;
7450                    ;       ERROR1: 2901 DATA PATH TEST ERROR (WORKING REG 0).
7450                    ;       ERROR2: 2901 DATA PATH TEST ERROR (WR CRR).
7450                    ;       ERROR3: 2901 DATA PATH TEST ERROR
7450                    ;
7450                    ;
7450                    ;       DEBUG:  FOR WRITE TO REG:
7450                    ;
7450                    ;               1) CHECK ''SRC CTL H'' FOR ''D.0'' = 7.
7450                    ;               2) CHECK ''ALU CTL H'' FOR ''R.OR.S'' = 3.
7450                    ;               3) CHECK ''DEST CTL H'' FOR ''WRITE.B.F'' = 3.
7450                    ;
7450                    ;               FOR READ OF REG:
7450                    ;
7450                    ;               4) CHECK ''SRC CTL H'' FOR ''0.A'' = 4.
7450                    ;               5) CHECK ''ALU CTL H'' FOR ''R.OR.S'' = 3.
7450                    ;               6) CHECK ''DEST CTL H'' FOR ''NOP'' = 1.
```

```
7450 1232                                      ;       7) CHECK "SRC CTL H,DEST CTL H,ALU CTL H" FOR CSR 22 - 14.
7450 1233                                      ;       8) B ADDRESS 0.
7450 1234                                      ;
7450 1235                                      ;       ERROR 2: CHECK WRITE CRR L DECODER
7450 1236                                      ;       ERROR 3: CHECK Y-BUS LATCH CLOCK LOGIC
7450 1237                                      ;
7450 1238                                      ;******************************************************************************
7450 1239
7450 1240
7450 1241
7450 1242 0000'CD 09 3E                       TEST9:          HEAD    <^X09>
          0F 0000'3A
          C9 045D'D2
          0F 0000'3A
          3D D3 0515'DA
7466 1243
7466 1244 00'0E'21                            1$:             LXI     H,MOV_CCR_WRO   ;LOAD THE CSR WITH A MOVE INSTRUCTION
7469 1245 029C'CD                                             CALL    LD_CSR_INST     ; USED TO PASS DATA THROUGH THE ALU.
746C 1246
746C 1247 0054'3A                                             LDA     ZERO_DAT        ;LOAD THE FIRST DATA PATTERN INTO A
746F 1248
746F 1249 02DB'CD                                             CALL    TEST9_1_0
7472 1250
7472 1251 0466'DA                                             JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7475 1252
7475 1253 00'0E'21                            2$:             LXI     H,MOV_CCR_WRO   ;LOAD THE CSR WITH A MOVE INSTRUCTION
7478 1254 029C'CD                                             CALL    LD_CSR_INST     ; USED TO PASS DATA THROUGH THE ALU.
747B 1255
747B 1256 0059'3A                                             LDA     ONE_DAT         ;LOAD ALL ZEROS INTO A
747E 1257
747E 1258 02DB'CD                                             CALL    TEST9_1_0
7481 1259
7481 1260 0475'DA                                             JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7484 1261
7484 1262
7484 1263
7484 1264 0058'3A                                             LDA     ONE_SHIFT+3     ;LOAD A ONE THRIUGH A ZERO FIELD
7487 1265 0087'32                                             STA     TEMPA_DAT       ; PATTERN INTO TEMPA_DAT.
748A 1266
748A 1267 C5 46 00'00'21        ******************            DO      8
          77 08 3E              *
7492 1268                       *
7492 1269 0307'CD               *                             CALL    TEST9_SHIFT
7495 1270                       *
7495 1271 0087'3A               *                             LDA     TEMPA_DAT       ;ROTATE THE TEST PATTERN TO THE LEFT
7498 1272 07                    *                             RLC                     ; TO SHIFT THE ONE THROUGH THE ZERO
7499 1273 0087'32               *                             STA     TEMPA_DAT       ; FIELD.
749C 1274                       *
749C 1275 35 00'00'21           ******************            ENDDO
          70 C1 0492'C2
74A5 1276
74A5 1277
74A5 1278 005D'3A                                             LDA     ZERO_SHIFT+3    ;LOAD A ZERO THROUGH A ONE FIELD
74A8 1279 0087'32                                             STA     TEMPA_DAT       ; PATTERN INTO TEMPA_DAT.
74AB 1280
74AB 1281 C5 46 00'00'21        ******************            DO      8
          77 08 3E              *
74B3 1282                       *
74B3 1283 0307'CD               *                             CALL    TEST9_SHIFT
74B6 1284                       *
```

```
74B6 1285 0087'3A           *                                    LDA     TEMPA_DAT        ;ROTATE THE TEST PATTERN TO THE LEFT
74B9 1286 07                *                                    RLC                      ; TO SHIFT THE ZERO THROUGH THE ONE
74BA 1287 0087'32           *                                    STA     TEMPA_DAT        ; FIELD.
74BD 1288                   *
74BD 1289 35 00'00'21       ******************                   ENDDO
          70 C1 04B3'C2
74C6 1290
74C6 1291
74C6 1292 0054'3A                                 3$:            LDA     ZERO_DAT         ;LOAD ALL ZEROS INTO A
74C9 1293
74C9 1294 033F'CD                                                CALL    TEST9_2_0
74CC 1295
74CC 1296 04C6'DA                                                JC      3$               ;...JUMP TO BEGINNING OF ERROR LOOP
74CF 1297
74CF 1298
74CF 1299 0059'3A                                 4$:            LDA     ONE_DAT          ;LOAD ALL ONES INTO A
74D2 1300
74D2 1301 033F'CD                                                CALL    TEST9_2_0
74D5 1302
74D5 1303 04CF'DA                                                JC      4$               ;...JUMP TO BEGINNING OF ERROR LOOP
74D8 1304
74D8 1305
74D8 1306 00'0E'21                                5$:            LXI     H,MOV_CCR_WR0    ;LOAD THE CSR WITH A MOVE INSTRUCTION
74DB 1307 029C'CD                                                CALL    LD_CSR_INST      ; USED TO PASS DATA THROUGH THE ALU
74DE 1308
74DE 1309 0054'3A                                                LDA     ZERO_DAT
74E1 1310
74E1 1311 CC D3                                                  OUT     WRITE            ;WRITE PATTERN TO THE D-BUS
74E3 1312
74E3 1313 23 D3                                                  OUT     SETSS            ;CLOCK DATA PATH
74E5 1314 22 D3                                                  OUT     CLRSS
74E7 1315
74E7 1316 00'00'21                                               LXI     H,X_MOV_WRWR     ;LOAD THE CSR WITH AN INSTRUCTION
74EA 1317 029C'CD                                                CALL    LD_CSR_INST      ; TO ENABLE THE WR TO BE READ
74ED 1318
74ED 1319 23 D3                                                  OUT     SETSS            ;CLOCK DATA PATH
74EF 1320 22 D3                                                  OUT     CLRSS
74F1 1321
74F1 1322 80 DB                                                  IN      READ             ;READ THE Y-BUS CONTENTS
74F3 1323
74F3 1324 0095'32                                                STA     REC_DAT          ;STORE THE Y-BUS CONTENTS IN REC_DAT
74F6 1325
74F6 1326 00'90'21          ******************                   IFNEQ   EXP_DAT,REC_DAT,1     ;COMPARE THE EXPECTED AND RECEIVED
          01 0E 00'95'11    *
          050C'CA 0000'CD   *
7504 1327                   *                                                                  ; DATA
7504 1328                   *
7504 1329 03 3E             *                                    MVI     A,3              ;ERROR NUMBER 3
7506 1330 0000'32           *                                    STA     CON_ERR_NUM
7509 1331                   *
7509 1332 0380'CD           *                                    CALL    TEST9_ERR2_3     ;REPORT ERROR
750C 1333                   *
750C 1334                   ******************                   ENDIF
750C 1335
750C 1336 0000'3A                                                LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
750F 1337 0F                                                     RRC
7510 1338 04D8'DA                                                JC      5$               ;...JUMP TO BEGINNING OF ERROR LOOP
7513 1339 3C D3                                                  TAIL
7515 1340
7515 1341
```

```
;*******************************************************************************
:
:
:        TESTA           2901 SHIFT FUNCTION TEST
:        -----           ------------------------
:
:                A ONE WILL BE PASSED TO THE 2901 IN BIT POSITION ZERO
:        AND SHIFTED TO THE LEFT 4 BITS THEN TO THE RIGHT 4 BITS AND
:        READ BACK. THIS WILL THEN BE FOLLOWED BY ANOTHER WRITE TO THE
:        2901 IN THE SAME MANNER EXECPT THE SHIFT WILL BE 8 BITS, THEN
:        AGAIN WITH 12 BITS AND SO ON UP TO THE 32 BITS OF THE 2901.
:        THIS WILL GIVE ISOLATION TO THE 4 BIT SLICE OF THE 2901 THAT
:        MIGHT FAIL TO SHIFT.
:
:        NOTE: DATA WILL BE READ FROM THE LOWER 8 BITS TO VERIFY THAT
:              THE DATA WAS IN FACT SHIFTED OUT OF BIT POSITION 0
:
:
:        TEST STEPS:
:
:                 1) LOAD 2901 WORKING REGISTEP WITH BIT 0 ON.
:                 2) SHIFT LEFT 8 TIMES.
:                 3) READ LOWER 8 BITS TO VERIFY DATA SHIFTED AWAY.
:                 4) LOAD 2901 WORKING REGISTER WITH BIT 0 ON.
:                 5) SHIFT LEFT N TIMES.
:                 6) READ LOWER 4 BITS TO VERIFY DATA SHIFTED AWAY.
:                 7) SHIFT RIGHT N TIMES.
:                 8) READ BACK DATA.
:                 9) PRINT ERROR IF ANY.
:                10) INC N BY 4 .
:                11) REPEAT 4-10 UNTIL 32 BITS OF SHIFT HAVE BEEN DONE.
:
:        ASSUMPTIONS:    THE DATA PATH TO AND FROM THE 2901 IS WORKING AS
:                        TESTED IN PREVIOUS TESTS.
:
:        ERROR1: DATA NOT SHIFTING AWAY.
:        ERROR2: DATA NOT BEING SHIFTED BACK CORRECTLY.
:
:        NOTE: ADDITIONAL DATA WILL GIVE THE SHIFT DEPTH (4,8,12...).
:
:
:        DEBUG:  1) CHECK THAT EACH BIT SLICE CAN SHIFT.  FOR AN ERROR
:                   IN SHIFTING AT ANY BIT POSITION N AN ERROR COULD OCCUR
:                   IN THE SERIAL IN LINES CONNECTING THE BIT SLICES OR THE
:                   SERIAL SHIFT OF THE PREVIOUS BIT SLICE.
:                2) ON ROL INSTRUCTIONS
:                   "SRC CTL H" FOR "0.B" = 3
:                   "ALU CTL H" FOR "R.OR.S" = 3
:                   "DEST CTL H" FOR "LSHF.RAM" = 7
:                3) ON ROR INSTRUCTIONS
:                   "SRC CTL H" FOR "0.B" = 3
:                   "ALU CTL H" FOR "R.OR.S" = 3
:                   "DEST CTL H" FOR "RSHF.RAM" = 5
:                4) CHECK "SRC CTL H,DEST CTL H,ALU CTL H" FOR CSR 22 - 14
:
:
;*******************************************************************************
```

```
7515 1402 0000'CD 0A 3E                         TESTA:          HEAD    <^X0A>
          0F 0000'3A
          C9 0522'D2
          0F 0000'3A
          3D D3 0609'DA
752B 1403
752B 1404 01 3E                                                 MVI     A,1             ;FIRST PART OF TEST A.
752D 1405 0000'32                                               STA     CON_ERR_NUM
7530 1406
7530 1407 08 3E                                                 MVI     A,8             ;SET SHIFT_CNT_DAT TO 8.
7532 1408 0008'32                                               STA     SHIFT_CNT_DAT
7535 1409
7535 1410 0090'32 AF                                            CLEAR   EXP_DAT         ;EXPECTED DATA IS ZERO AS DATA
7539 1411                                                                               ; SHOULD GET SHIFTED AWAY.
7539 1412
7539 1413 00'0E'21                              1$:             LXI     H,MOV_CCR_WRO   ;LOAD THE CSR WITH AN INSTRUCTION TO
753C 1414 029C'CD                                               CALL    LD_CSR_INST     ; ENABLE THE 2901 TO BE LOADED.
753F 1415
753F 1416 01 3E                                                 MVI     A,1             ;LOAD A ONE INTO A.
7541 1417 CC D3                                                 OUT     WRITE           ;WRITE THE DATA TO THE D-BUS LATCH.
7543 1418
7543 1419 23 D3                                                 OUT     SETSS           ;LOAD THE DATA INTO THE 2901'S WRO.
7545 1420 22 D3                                                 OUT     CLRSS
7547 1421
7547 1422 0008'3A                                               LDA     SHIFT_CNT_DAT   ;LOAD THE SHIFT COUNTER DATA INTO THE
754A 1423 0007'32                                               STA     SHIFT_CNT       ; SHIFT COUNTER.
754D 1424
754D 1425 0000'CD                               2$:             CALL    SHIFT_L_2901    ;SHIFT THE DATA TO THE LEFT ONE BIT.
7550 1426
7550 1427 00A9'CD                                               CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7553 1428
7553 1429 0007'3A                                               LDA     SHIFT_CNT       ;DECREMENT THE SHIFT COUNTER AND SHIFT
7556 1430 3D                                                    DCR     A               ; AGAIN IF APPROPRIATE.
7557 1431 0007'32                                               STA     SHIFT_CNT
755A 1432 054D'C2                                               JNZ     2$
755D 1433
755D 1434 00'00'21                                              LXI     H,X_MOV_WRWR    ;PREPARE TO READ THE LOW BYTE OF WRO.
7560 1435 029C'CD                                               CALL    LD_CSR_INST
7563 1436
7563 1437 EC D3                                                 OUT     YBUSRD          ;LOAD THE DATA INTO THE Y-BUS LATCH.
7565 1438 80 DB                                                 IN      READ            ;READ THE Y-BUS.
7567 1439 0095'32                                               STA     REC_DAT         ;STORE THE DATA IN REC_DAT.
756A 1440
756A 1441 00'90'21          ******************                  IFNEQ   EXP_DAT,REC_DAT,1   ;COMPARE THE EXPECTED AND
          01 0E 00'95'11    *
          057B'CA 0000'CD   *
7578 1442                   *                                                                   ; RECEIVED DATA.
7578 1443                   *
7578 1444 03A0'CD           *                                   CALL    TESTA_ERROR     ;PRINT AN ERROR.
757B 1445                   *
757B 1446                   ******************                  ENDIF
757B 1447
757B 1448 0000'3A                                               LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
757E 1449 0F                                                    RRC
757F 1450 0539'DA                                               JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7582 1451
7582 1452
7582 1453 02 3E                                                 MVI     A,2             ;SECOND PART OF TEST A.
7584 1454 0000'32                                               STA     CON_ERR_NUM
7587 1455
```

```
7587 1456 04 3E                                               MVI     A,4              ;SET SHIFT_CNT_DAT TO FOUR.
7589 1457 0008'32                                             STA     SHIFT_CNT_DAT
758C 1458
758C 1459 C5 46 00'00'21      ******************              DO      7                ;7 SHIFT DEPTHS ARE USED (4,8,12...28).
          77 07 3E            *
7594 1460                     *
7594 1461 00'0E'21            *                      3$:      LXI     H,MOV_CCR_WRO    ;LOAD THE CSR WITH AN INSTRUCTION TO
7597 1462 029C'CD             *                               CALL    LD_CSR_INST      ; ENABLE THE 2901 TO BE LOADED.
759A 1463                     *
759A 1464 01 3E               *                               MVI     A,1              ;LOAD A ONE INTO A AND INTO EXP_DAT.
759C 1465 0090'32             *                               STA     EXP_DAT
759F 1466 CC D3               *                               OUT     WRITE            ;WRITE THE DATA TO THE D-BUS LATCH.
75A1 1467                     *
75A1 1468 23 D3               *                               OUT     SETSS            ;LOAD THE DATA INTO THE 2901'S WRO.
75A3 1469 22 D3               *                               OUT     CLRSS
75A5 1470                     *
75A5 1471 0008'3A             *                               LDA     SHIFT_CNT_DAT    ;LOAD THE SHIFT COUNTER DATA INTO THE
75A8 1472 0007'32             *                               STA     SHIFT_CNT        ; SHIFT COUNTER.
75AB 1473                     *
75AB 1474 0000'CD             *                      4$:      CALL    SHIFT_L_2901     ;SHIFT THE DATA TO THE LEFT ONE BIT.
75AE 1475                     *
75AE 1476 00A9'CD             *                               CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
75B1 1477                     *
75B1 1478 0007'3A             *                               LDA     SHIFT_CNT        ;DECREMENT THE SHIFT COUNTER AND SHIFT
75B4 1479 3D                  *                               DCR     A                ; AGAIN IF APPROPRIATE.
75B5 1480 0007'32             *                               STA     SHIFT_CNT
75B8 1481 05AB'C2             *                               JNZ     4$
75BB 1482                     *
75BB 1483 0008'3A             *                               LDA     SHIFT_CNT_DAT    ;LOAD THE SHIFT COUNTER DATA INTO THE
75BE 1484 0007'32             *                               STA     SHIFT_CNT        ; SHIFT COUNTER.
75C1 1485                     *
75C1 1486 00'01'21            *                      5$:      LXI     H,X_SHIFT_R+1    ;PERFORM THE INSTRUCTION USED TO SHIFT
75C4 1487 0294'CD             *                               CALL    EXEC_INST        ; WR[0] TO THE RIGHT ONE PLACE.
75C7 1488                     *
75C7 1489 0007'3A             *                               LDA     SHIFT_CNT        ;DECREMENT THE SHIFT COUNTER AND SHIFT
75CA 1490 3D                  *                               DCR     A                ; AGAIN IF APPROPRIATE.
75CB 1491 0007'32             *                               STA     SHIFT_CNT
75CE 1492 05C1'C2             *                               JNZ     5$
75D1 1493                     *
75D1 1494 00'00'21            *                               LXI     H,X_MOV_WRWR     ;PREPARE TO READ THE DATA BACK.
75D4 1495 029C'CD             *                               CALL    LD_CSR_INST
75D7 1496                     *
75D7 1497 EC D3               *                               OUT     YBUSRD           ;LOAD THE DATA INTO THE Y-BUS LATCH.
75D9 1498 80 DB               *                               IN      READ             ;READ THE Y-BUS.
75DB 1499 0095'32             *                               STA     REC_DAT          ;STORE THE DATA IN REC_DAT.
75DE 1500                     *
75DE 1501 00'90'21            * ****************              IFNEQ   EXP_DAT,REC_DAT,1    ;COMPARE THE EXPECTED AND
          01 0E 00'95'11      * *
          05EF'CA 0000'CD     * *
75EC 1502                     * *                                                          ; RECEIVED DATA.
75EC 1503                     * *
75EC 1504 03A0'CD             * *                             CALL    TESTA_ERROR      ;PRINT AN ERROR.
75EF 1505                     * *
75EF 1506                     * ****************              ENDIF
75EF 1507                     *
75EF 1508 0000'3A             *                               LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
75F2 1509 0F                  *                               RRC
75F3 1510 0594'DA             *                               JC      3$               ;...JUMP TO BEGINNING OF ERROR LOOP.
75F6 1511                     *
75F6 1512 0008'3A             *                               LDA     SHIFT_CNT_DAT    ;ADD 4 TO THE VALUE OF SHIFT_CNT_DAT
```

```
75F9 1513 04 C6                *                   ADI     4                ; TO SHIFT THE DATA FOUR BITS DEEPER
75FB 1514 0008'32              *                   STA     SHIFT_CNT_DAT    ; INTO THE 2901 ON THE NEXT PASS.
75FE 1515                      *
75FE 1516 35 00'00'21          ******************  ENDDO
          70 C1 0594'C2
7607 1517
7607 1518 3C D3                                    TAIL
7609 1519
7609 1520
7609 1521                 ;*****************************************************************************
7609 1522                 ;
7609 1523                 ;
7609 1524                 ;       TESTB           WORKING REGISTER CLEAR TEST
7609 1525                 ;       -----           ---------------------------
7609 1526                 ;
7609 1527                 ;               THIS TEST WILL LOAD A WORKING REGISTER WITH EACH OF THE
7609 1528                 ;       STANDARD PATTERNS OF ONE RIPPLED THROUGH A FIELD OF ZEROS AND A
7609 1529                 ;       ZERO RIPPLED THROUGH A FIELD OF ONES.  THE REGISTER WILL THEN BE
7609 1530                 ;       CLEARED AND READ TO BE SURE IT IS CLEARED. BECAUSE THE READS AND
7609 1531                 ;       WRITES OF WRO USED SUPPORT MICROCODE, THE FIRST PART OF THIS TEST
7609 1532                 ;       PERFORMS A FAST WRITE WRO VERIFICATION. THE SUPPORT CODE IS SINGLE
7609 1533                 ;       STEPPED THROUGH AND THE VALUE OF THE UPC IS CHECKED APTER EACH
7609 1534                 ;       INSTRUCTION.
7609 1535                 ;
7609 1536                 ;
7609 1537                 ;       TEST STEPS:
7609 1538                 ;
7609 1539                 ;                1) STEP THROUGH SUPPORT MICROCODE.
7609 1540                 ;                2) COMPARE EXPECTED AND ACTUAL UPC VALUE AFTER EACH STEP.
7609 1541                 ;                3) PRINT ERROR IF ANY.
7609 1542                 ;                4) EXECUTE A CLEAR WRO AND THEN A COMPLEMENT WRO INSTRUCTION,
7609 1543                 ;                   READ WRO AND CHECK FOR ALL ONES.
7609 1544                 ;                5) WRITE ALL ONES TO WRO AND THEN READ WRO AND CHECK FOR ALL
7609 1545                 ;                   ONES.
7609 1546                 ;                6) LOAD WR WITH DATA PATTERN.
7609 1547                 ;                7) PUT CLR WR INST IN CSR AND EXECUTE.
7609 1548                 ;                8) READ THE WORKING REGISTER TO PROVE IT WAS CLEARED.
7609 1549                 ;                9) PRINT ERROR IF ANY.
7609 1550                 ;               10) REPEAT 6-9 FOR ALL DATA PATTERNS.
7609 1551                 ;
7609 1552                 ;
7609 1553                 ;       ERROR1: VERIFY FAST WRITE FAILURE. ADDITIONAL DATA WILL SPECIFY
7609 1554                 ;               WHETHER CPU ATTN IS CLEAR: ADD_DAT=0, OR SET: ADD_DAT=1.
7609 1555                 ;       ERROR2: WORKING REGISTER DOES NOT CONTAIN ALL ONES AFTER CLEAR,
7609 1556                 ;               COMPLEMENT INSTRUCTION SEQUENCE.
7609 1557                 ;       ERROR3: WORKING REGISTER DOES NOT CONTAIN ALL ONES AFTER WRITING
7609 1558                 ;               ALL ONES TO IT.
7609 1559                 ;       ERROR4: CLEAR WORKING REGISTER TEST FAILURE. FOR THIS ERROR,
7609 1560                 ;               ADDITIONAL DATA WILL CONTAIN THE DATA SENT TO WRO.
7609 1561                 ;
7609 1562                 ;
7609 1563                 ;       DEBUG:  1) CLR WR[] USES 'WR.XOR.WR'
7609 1564                 ;                  ''SRC CTL H'' FOR 'A.B'' = 1
7609 1565                 ;                  ''ALU CTL H'' FOR 'R.XOR.S'' = 6
7609 1566                 ;                  ''DEST CTL H'' FOR 'WRITE B.F'' = 3
7609 1567                 ;               2) CHECK ''SRC CTL H,DEST CTL H,ALU CTL H'' FOR CSR 22 - 14
7609 1568                 ;
7609 1569                 ;
7609 1570                 ;*****************************************************************************
7609 1571
```

```
7609 1572
7609 1573 0000'CD 0B 3E                          TESTB:          HEAD    <^X0B>
          0F 0000'3A
          C9 0616'D2
          0F 0000'3A
          3D D3 083E'DA
761F 1574
761F 1575 00B9'CD                                                CALL    LD_SUPPORT      ;LOAD SUPPORT CODE FOR FAST READS AND
7622 1576                                                                                ; WRITES OF WRO.
7622 1577
7622 1578
7622 1579 01 3E                                                  MVI     A,1             ;FIRST PART OF TEST B.
7624 1580 0000'32                                                STA     CON_ERR_NUM
7627 1581
7627 1582 04 0E 00'00'21                                         ZERO    CON_ADD_DAT,4   ;CLEAR ADDITIONAL DATA.
          0D 23 77 AF
          062D'C2
7633 1583
7633 1584 FF FF 21                                               LXI     H,<^XFFFF>      ;LAST TWO BYTES OF EXPECTED AND
7636 1585 0000'22                                                SHLD    CON_MSK_DAT     ; RECEIVED DATA ARE OF INTEREST.
7639 1586 02 0E 00'02'21                                         ZERO    CON_MSK_DAT+2,2
          0D 23 77 AF
          063F'C2
7645 1587
7645 1588 00 00 21                               1$:             LXI     H,0             ;LOAD A ZERO INTO THE UPC.
7648 1589 0000'22                                                SHLD    UPC_VALUE
764B 1590 0000'CD                                                CALL    LOAD_UPC
764E 1591
764E 1592 0000'CD                                                CALL    NOP_CSR         ;LOAD THE CSR WITH A NOP.
7651 1593
7651 1594 00'44'21                                               LXI     H,VER_WRITE_TAB ;POINT TO THE TOP OF THE VERIFY
7654 1595 0000'22                                                SHLD    WRITE_TAB_PNT   ; WRITE TABLE.
7657 1596
7657 1597 AB D3                                                  OUT     CLRATN          ;CLEAR CPU ATTENTION.
7659 1598
7659 1599 C5 46 00'00'21    ******************                   DO      5
          77 05 3E          *
7661 1600                   *
7661 1601 00A9'CD           *                                    CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7664 1602                   *
7664 1603 0000'CD           *                                    CALL    MICRO_STEP_CPU  ;EXECUTE ONE MICROINSTRUCTION.
7667 1604 011B'CD           *                                    CALL    SET_COMPARE     ;READ THE UPC AND PUT IN REC_DAT.
766A 1605                   *                                                           ;ALSO, PUT THE CORRESPONDING WRITE
766A 1606                   *                                                           ; TABLE ENTRY IN EXP_DAT.
766A 1607                   *
766A 1608 00'92'21          * ****************                   IFNEQ   EXP_DAT+2,REC_DAT+2,2   ;COMPARE THE EXPECTED AND
          02 0E 00'97'11    * *
          067B'CA 0000'CD   * *
7678 1609                   * *                                                                 ; RECEIVED DATA.
7678 1610                   * *
7678 1611 03CD'CD           * *                                  CALL    TESTB_ERROR     ;PRINT AN ERROR MESSAGE.
767B 1612                   * *
767B 1613                   * ****************                   ENDIF
767B 1614                   *
767B 1615 35 00'00'21       ******************                   ENDDO
          70 C1 0661'C2
7684 1616
7684 1617 0000'3A                                                LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7687 1618 0F                                                     RRC
7688 1619 0645'DA                                                JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
```

```
768B 1620
768B 1621 00'4E'21                   2$:         LXI     H,VER_WRITE_TAB+10      ;POINT TO THE PROPER TABLE
768E 1622 0000'22                                SHLD    WRITE_TAB_PNT           ; ENTRY TO COMPARE THE UPC.
7691 1623
7691 1624 AA D3                                  OUT     SETATN          ;SET CPU ATTENTION.
7693 1625
7693 1626 C5 46 00'00'21     *******************             DO      3
          77 03 3E           *
769B 1627                    *
769B 1628 00A9'CD            *                   CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
769E 1629                    *
769E 1630 0000'CD            *                   CALL    MICRO_STEP_CPU  ;EXECUTE ONE MICROINSTRUCTION.
76A1 1631 011B'CD            *                   CALL    SET_COMPARE     ;READ THE UPC AND PUT IN REC_DAT.
76A4 1632                    *                                           ;ALSO, PUT THE CORRESPONDING WRITE
76A4 1633                    *                                           ; TABLE ENTRY IN EXP_DAT.
76A4 1634                    *
76A4 1635 00'92'21           * ****************                  IFNEQ   EXP_DAT+2,REC_DAT+2,2   ;COMPARE THE EXPECTED AND
          02 0E 00'97'11     * *
          06BA'CA 0000'CD    * *
76B2 1636                    * *                                                         ; RECEIVED DATA.
76B2 1637                    * *
76B2 1638 0003'32 3C AF      * *                 SET     CON_ADD_DAT+3   ;ADDITIONAL DATA INDICATES CPU ATTN
76B7 1639                    * *                                         ; WAS SET WHEN ERROR OCCURRED.
76B7 1640                    * *
76B7 1641 03CD'CD            * *                 CALL    TESTB_ERROR     ;PRINT AN ERROR MESSAGE.
76BA 1642                    * *
76BA 1643                    * ****************                  ENDIF
76BA 1644                    *
76BA 1645 35 00'00'21        *******************             ENDDO
          70 C1 069B'C2
76C3 1646
76C3 1647 0000'3A                                LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
76C6 1648 0F                                     RRC
76C7 1649 068B'DA                                JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
76CA 1650
76CA 1651
76CA 1652
76CA 1653 02 3E                                  MVI     A,2             ;SECOND PART OF TESTB.
76CC 1654 0000'32                                STA     CON_ERR_NUM
76CF 1655
76CF 1656 04 0E 00'00'21                         ZERO    CON_MSK_DAT,4   ;ALL OF EXPECETED AND RECEIVED
          0D 23 77 AF
          06D5'C2
76DB 1657                                                                ; DATA IS OF INTEREST.
76DB 1658
76DB 1659 00'59'21                               LXI     H,ONE_DAT       ;LOAD ALL ONES INTO EXP_DAT.
76DE 1660 00'90'11                               LXI     D,EXP_DAT
76E1 1661 0000'CD                                CALL    MOVER_4
76E4 1662
76E4 1663 00'01'21                   3$:         LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
76E7 1664 0294'CD                                CALL    EXEC_INST
76EA 1665
76EA 1666 00'00'21                               LXI     H,X_COM_WR0     ;COMPLEMENT WR0.
76ED 1667 0294'CD                                CALL    EXEC_INST
76F0 1668
76F0 1669 01B9'CD                                CALL    FAST_READ_WR0   ;READ BACK THE CONTENTS OF WR0.
76F3 1670
76F3 1671 00'00'21                               LXI     H,DATA0         ;MOVE THE DATA IN DATA0 TO REC_DAT.
76F6 1672 00'95'11                               LXI     D,REC_DAT
76F9 1673 0000'CD                                CALL    MOVER_4
```

```
76FC 1674
76FC 1675 00'90'21             ******************          IFNEQ   EXP_DAT,REC_DAT,4     ;COMPARE THE EXPECTED AND
          04 0E 00'95'11       *
          0712'CA 0000'CD      *
770A 1676                      *                                                     ; RECEIVED DATA.
770A 1677                      *
770A 1678 0000'32 3C AF        *                           SET     NA_OTHER        ;NO ADDITIONAL DATA.
770F 1679                      *
770F 1680 03CD'CD              *                           CALL    TESTB_ERROR     ;PRINT AN ERROR.
7712 1681                      *
7712 1682                      ******************          ENDIF
7712 1683
7712 1684 00A9'CD                                          CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7715 1685
7715 1686 0000'3A                                          LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7718 1687 0F                                               RRC
7719 1688 06E4'DA                                          JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
771C 1689
771C 1690
771C 1691
771C 1692
771C 1693 03 3E                                            MVI     A,3             ;THIRD PART OF TESTB.
771E 1694 0000'32                                          STA     CON_ERR_NUM
7721 1695
7721 1696 00'59'21                                         LXI     H,ONE_DAT       ;LOAD ALL ONE INTO EXP_DAT.
7724 1697 00'90'11                                         LXI     D,EXP_DAT
7727 1698 0000'CD                                          CALL    MOVER_4
772A 1699
772A 1700 00'59'21                                4$:      LXI     H,ONE_DAT       ;LOAD ALL ONES INTO DATA0.
772D 1701 00'00'11                                         LXI     D,DATA0
7730 1702 0000'CD                                          CALL    MOVER_4
7733 1703
7733 1704 0172'CD                                          CALL    FAST_WRITE_WR0  ;WRITE THIS TEST PATTERN INTO WR0.
7736 1705
7736 1706 01B9'CD                                          CALL    FAST_READ_WR0   ;READ BACK THE CONTENTS OF WR0.
7739 1707
7739 1708 00'00'21                                         LXI     H,DATA0         ;MOVE THE DATA IN DATA0 TO REC_DAT.
773C 1709 00'95'11                                         LXI     D,REC_DAT
773F 1710 0000'CD                                          CALL    MOVER_4
7742 1711
7742 1712 00'90'21             ******************          IFNEQ   EXP_DAT,REC_DAT,4     ;COMPARE THE EXPECTED AND
          04 0E 00'95'11       *
          0758'CA 0000'CD      *
7750 1713                      *                                                     ; RECEIVED DATA.
7750 1714                      *
7750 1715 0000'32 3C AF        *                           SET     NA_OTHER        ;NO ADDITIONAL DATA.
7755 1716                      *
7755 1717 03CD'CD              *                           CALL    TESTB_ERROR     ;PRINT AN ERROR.
7758 1718                      *
7758 1719                      ******************          ENDIF
7758 1720
7758 1721 00A9'CD                                          CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
775B 1722
775B 1723 0000'3A                                          LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
775E 1724 0F                                               RRC
775F 1725 072A'DA                                          JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7762 1726
7762 1727
7762 1728
7762 1729
```

```
ZZ-ENKBB-2.0    7000    4
7762 1730 04 3E                                               MVI     A,4                 ;FOURTH PART OF TESTB.
7764 1731 0000'32                                             STA     CON_ERR_NUM
7767 1732
7767 1733 04 0E 00'90'21                                      ZERO    EXP_DAT,4           ;EXPECTED DATA IS ALL ZEROS.
          0D 23 77 AF
          076D'C2
7773 1734
7773 1735 00'55'21                                            LXI     H,ONE_SHIFT         ;LOAD A ONE THROUGH A ZERO FIELD
7776 1736 00'8B'11                                            LXI     D,TEMPB_DAT         ; PATTERN INTO TEMPB_DAT.
7779 1737 0000'CD                                            CALL    MOVER_4
777C 1738
777C 1739 C5 46 00'00'21      ******************              DO      32                  ;32 DIFFERENT TEST PATTERNS.
          77 20 3E            *
7784 1740                     *
7784 1741 00'8B'21            *                       5$:     LXI     H,TEMPB_DAT         ;MOVE THE PATTERN TO DATA0.
7787 1742 00'00'11            *                               LXI     D,DATA0
778A 1743 0000'CD             *                               CALL    MOVER_4
778D 1744                     *
778D 1745 0172'CD             *                               CALL    FAST_WRITE_WR0  ;WRITE THE PATTERN TO WR0
7790 1746                     *
7790 1747 00'01'21            *                               LXI     H,X_CLR_WR0+1       ;CLEAR  WR0.
7793 1748 0294'CD             *                               CALL    EXEC_INST
7796 1749                     *
7796 1750 01B9'CD             *                               CALL    FAST_READ_WR0       ;READ THE DATA FROM WR0.
7799 1751                     *
7799 1752 00'00'21            *                               LXI     H,DATA0             ;MOVE THE RETURNED DATA TO REC_DAT.
779C 1753 00'95'11            *                               LXI     D,REC_DAT
779F 1754 0000'CD             *                               CALL    MOVER_4
77A2 1755                     *
77A2 1756 00'90'21            * *****************             IFNEQ   EXP_DAT,REC_DAT,4       ;COMPARE THE EXPECTED AND
          04 0E 00'95'11      * *
          07BC'CA 0000'CD     * *
77B0 1757                     * *                                                             ; RECEIVED DATA.
77B0 1758                     * *
77B0 1759 00'8B'21            * *                             LXI     H,TEMPB_DAT         ;ADDITIONAL DATA CONTAINS DATA
77B3 1760 00'00'11            * *                             LXI     D,CON_ADD_DAT       ; SENT TO WR0.
77B6 1761 0000'CD             * *                             CALL    MOVER_4
77B9 1762                     * *
77B9 1763 03CD'CD             * *                             CALL    TESTB_ERROR         ;PRINT AN ERROR.
77BC 1764                     * *
77BC 1765                     * *****************             ENDIF
77BC 1766                     *
77BC 1767 0000'3A             *                               LDA     ERROR_CON           ;IF THE ERROR LOOPING FLAG IS SET...
77BF 1768 0F                  *                               RRC
77C0 1769 0784'DA             *                               JC      5$                  ;...JUMP TO BEGINNING OF ERROR LOOP.
77C3 1770                     *
77C3 1771 B7                  *                               ORA     A                   ;CLEAR THE CARRY.
77C4 1772                     *
77C4 1773 00'8A'21            *                               LXI     H,TEMPB_SHIFT       ;SHIFT THE ONE THROUGH THE ZERO FIELD.
77C7 1774 0000'22             *                               SHLD    SHIFT_PNT
77CA 1775 0000'CD             *                               CALL    SHIFTER
77CD 1776                     *
77CD 1777 35 00'00'21         ******************              ENDDO
          70 C1 0784'C2
77D6 1778
77D6 1779 00A9'CD                                             CALL    CHECK_RD            ;CALL BOOT CODE IF UNDER RD CONTROL
77D9 1780
77D9 1781
77D9 1782
77D9 1783
```

```
77D9 1784 00'5A'21                                             LXI     H,ZERO_SHIFT    ;LOAD A ZERO THROUGH A ONE FIELD
77DC 1785 00'8B'11                                             LXI     D,TEMPB_DAT     ; PATTERN INTO TEMPB_DAT.
77DF 1786 0000'CD                                              CALL    MOVER_4
77E2 1787
77E2 1788 C5 46 00'00'21      ******************               DO      32
          77 20 3E            *
77EA 1789                     *
77EA 1790 00'8B'21            *                  6$:           LXI     H,TEMPB_DAT     ;LOAD THE PATTERN INTO DATA0.
77ED 1791 00'00'11            *                                LXI     D,DATA0
77F0 1792 0000'CD             *                                CALL    MOVER_4
77F3 1793                     *
77F3 1794 0172'CD             *                                CALL    FAST_WRITE_WR0  ;WRITE THE PATTERN TO WR0
77F6 1795                     *
77F6 1796 00'01'21            *                                LXI     H,X_CLR_WR0+1   ;CLEAR  WR0.
77F9 1797 0294'CD             *                                CALL    EXEC_INST
77FC 1798                     *
77FC 1799 01B9'CD             *                                CALL    FAST_READ_WR0   ;READ THE DATA FROM WR0.
77FF 1800                     *
77FF 1801 00'00'21            *                                LXI     H,DATA0         ;MOVE THE RETURNED DATA TO REC_DAT.
7802 1802 00'95'11            *                                LXI     D,REC_DAT
7805 1803 0000'CD             *                                CALL    MOVER_4
7808 1804                     *
7808 1805 00'90'21            * ****************               IFNEQ   EXP_DAT,REC_DAT,4    ;COMPARE THE EXPECTED AND
          04 0E 00'95'11      * *
          0822'CA 0000'CD     * *
7816 1806                     * *                                                           ; RECEIVED DATA.
7816 1807                     * *
7816 1808 00'8B'21            * *                              LXI     H,TEMPB_DAT     ;ADDITIONAL DATA CONTAINS DATA
7819 1809 00'00'11            * *                              LXI     D,CON_ADD_DAT   ; SENT TO WR0.
781C 1810 0000'CD             * *                              CALL    MOVER_4
781F 1811                     * *
781F 1812 03CD'CD             * *                              CALL    TESTB_ERROR     ;PRINT AN ERROR.
7822 1813                     * *
7822 1814                     * ****************               ENDIF
7822 1815                     *
7822 1816 0000'3A             *                                LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7825 1817 0F                  *                                RRC
7826 1818 07EA'DA             *                                JC      6$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7829 1819                     *
7829 1820 37                  *                                STC                     ;SET THE CARRY.
782A 1821                     *
782A 1822 00'8A'21            *                                LXI     H,TEMPB_SHIFT   ;SHIFT THE ZERO THROUGH THE ONE STRING.
782D 1823 0000'22             *                                SHLD    SHIFT_PNT
7830 1824 0000'CD             *                                CALL    SHIFTER
7833 1825                     *
7833 1826 35 00'00'21         ******************               ENDDO
          70 C1 07EA'C2
783C 1827
783C 1828 3C D3                                                TAIL
783E 1829
783E 1830
783E 1831
783E 1832                                      ;******************************************************************************
783E 1833                                      ;
783E 1834                                      ;
783E 1835                                      ;       TESTC           WORKING REGISTER TEST (WORKING REG 1)
783E 1836                                      ;       -----           -------------------------------------
783E 1837                                      ;
783E 1838                                      ;               A STANDARD TEST PATTERN OF RIPPLING A ZERO THROUGH A
783E 1839                                      ;       FIELD OF ONES AND A ONE THROUGH A FIELD OF ZEROS WILL BE USED
```

```
783E 1840                                       ;       TO TEST THIS WORKING REGISTER FOR FUTURE USE. WRO WILL ALSO BE
783E 1841                                       ;       WRITTEN TO ASSURE THAT DUAL ADDRESSING DOES NOT OCCUR
783E 1842                                       ;
783E 1843                                       ;
783E 1844                                       ;       TEST STEPS:
783E 1845                                       ;
783E 1846                                       ;               1) WRITE DATA TO WORKING REG 1.
783E 1847                                       ;               2) READ DATA BACK.
783E 1848                                       ;               3) PRINT ERRORS IF ANY.
783E 1849                                       ;               4) REPEAT 1-3 FOR ALL PATTERNS.
783E 1850                                       ;               5) WRITE WRO WITH 1'S.
783E 1851                                       ;               6) WRITE WR1 WITH 0'S.
783E 1852                                       ;               7) READ 1'S IN WRO.
783E 1853                                       ;               8) READ 0'S IN WR1.
783E 1854                                       ;               9) PRINT ERRORS IF ANY.
783E 1855                                       ;
783E 1856                                       ;       ASSUMPTIONS:    THE DATA PATH TO AND FROM THE 2901 IS WORKING
783E 1857                                       ;                       AS TESTED IN PREVIOUS TESTS.
783E 1858                                       ;
783E 1859                                       ;
783E 1860                                       ;       ERROR1: 2901 WORKING REGISTER TEST FAILURE.
783E 1861                                       ;       ERROR2: DUAL ADDRESSING ERROR.
783E 1862                                       ;
783E 1863                                       ;       DEBUG:  FOR WRITE TO REG:
783E 1864                                       ;
783E 1865                                       ;               1) CHECK ''SRC CTL H'' FOR ''D.0'' = 7.
783E 1866                                       ;               2) CHECK ''ALU CTL H'' FOR ''R.OR.S'' = 3.
783E 1867                                       ;               3) CHECK ''DEST CTL H'' FOR ''WRITE.B.F'' = 3.
783E 1868                                       ;
783E 1869                                       ;               FOR READ OF REG:
783E 1870                                       ;
783E 1871                                       ;               4) CHECK ''SRC CTL H'' FOR ''0.A'' = 4.
783E 1872                                       ;               5) CHECK ''ALU CTL H'' FOR ''R.OR.S'' = 3.
783E 1873                                       ;               6) CHECK ''DEST CTL H'' FOR ''NOP'' = 1.
783E 1874                                       ;               7) CHECK ''SRC CTL H,DEST CTL H,ALU CTL H'' FOR CSR 22 - 14.
783E 1875                                       ;               8) SINCE THE PATH TO AND FROM THE 2901 HAS ALREADY
783E 1876                                       ;                  BEEN TESTED IT IS PROBABLY THE 2901 ITSELF OR
783E 1877                                       ;                  DUAL ADDRESSING.
783E 1878                                       ;               9) CHECK TAT B ADDRESS BIT 0 CHANGES.
783E 1879                                       ;
783E 1880                                       ;
783E 1881                                       ;*******************************************************************************
783E 1882
783E 1883
783E 1884
783E 1885 0000'CD 0C 3E                 TESTC:          HEAD    <^X0C>
          0F 0000'3A
          C9 084B'D2
          0F 0000'3A
          3D D3 0A19'DA
7854 1886
7854 1887 00B9'CD                                       CALL    LD_SUPPORT      ;LOAD SUPPORT CODE FOR FAST READS AND
7857 1888                                                                       ; WRITES OF WRO.
7857 1889
7857 1890 01 3E                                         MVI     A,1             ;FIRST PART OF TEST C.
7859 1891 0000'32                                       STA     CON_ERR_NUM
785C 1892
785C 1893 00'54'21                                      LXI     H,ZERO_DAT      ;LOAD ALL ZEROS INTO EXP_DAT.
785F 1894 00'90'11                                      LXI     D,EXP_DAT
7862 1895 0000'CD                                       CALL    MOVER_4
```

```
7865 1896
7865 1897 00'54'21                      1$:             LXI     H,ZERO_DAT      ;LOAD ALL ZEROS INTO DATA0.
7868 1898 00'00'11                                      LXI     D,DATA0
786B 1899 0000'CD                                       CALL    MOVER_4
786E 1900
786E 1901 0205'CD                                       CALL    WRITE_WR1_32    ;WRITE PATTERN TO WR[1].
7871 1902
7871 1903 023E'CD                                       CALL    READ_WR1_32     ;READ DATA BACK FROM WR[1].
7874 1904
7874 1905 00'00'21                                      LXI     H,DATA0         ;MOVE THE RECEIVED DATA TO REC_DAT.
7877 1906 00'95'11                                      LXI     D,REC_DAT
787A 1907 0000'CD                                       CALL    MOVER_4
787D 1908
787D 1909 00'90'21          ******************          IFNEQ   EXP_DAT,REC_DAT,4       ;COMPARE THE EXPECTED AND
          04 0E 00'95'11    *
          088E'CA 0000'CD   *
788B 1910                   *                                                           ; RECEIVED DATA.
788B 1911                   *
788B 1912 03E7'CD           *                           CALL    TESTC_ERROR     ;PRINT AN ERROR MESSAGE.
788E 1913                   *
788E 1914                   ******************          ENDIF
788E 1915
788E 1916 0000'3A                                       LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7891 1917 0F                                            RRC
7892 1918 0865'DA                                       JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7895 1919
7895 1920
7895 1921
7895 1922 00'59'21                                      LXI     H,ONE_DAT       ;LOAD ALL ONES INTO EXP_DAT.
7898 1923 00'90'11                                      LXI     D,EXP_DAT
789B 1924 0000'CD                                       CALL    MOVER_4
789E 1925
789E 1926 00'59'21                      2$:             LXI     H,ONE_DAT       ;LOAD ALL ONES INTO DATA0.
78A1 1927 00'00'11                                      LXI     D,DATA0
78A4 1928 0000'CD                                       CALL    MOVER_4
78A7 1929
78A7 1930 0205'CD                                       CALL    WRITE_WR1_32    ;WRITE PATTERN TO WR[1].
78AA 1931
78AA 1932 023E'CD                                       CALL    READ_WR1_32     ;READ DATA BACK FROM WR[1].
78AD 1933
78AD 1934 00'00'21                                      LXI     H,DATA0         ;MOVE RECEIVED DATA TO REC_DAT.
78B0 1935 00'95'11                                      LXI     D,REC_DAT
78B3 1936 0000'CD                                       CALL    MOVER_4
78B6 1937
78B6 1938 00'90'21          ******************          IFNEQ   EXP_DAT,REC_DAT,4       ;COMPARE THE EXPECTED AND
          04 0E 00'95'11    *
          08C7'CA 0000'CD   *
78C4 1939                   *                                                           ; RECEIVED DATA.
78C4 1940                   *
78C4 1941 03E7'CD           *                           CALL    TESTC_ERROR     ;PRINT AN ERROR MESSAGE.
78C7 1942                   *
78C7 1943                   ******************          ENDIF
78C7 1944
78C7 1945 0000'3A                                       LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
78CA 1946 0F                                            RRC
78CB 1947 089E'DA                                       JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
78CE 1948
78CE 1949
78CE 1950
78CE 1951 00'55'21                                      LXI     H,ONE_SHIFT     ;LOAD ONE THROUGH A ZERO FIELD
```

```
78D1 1952 00'8B'11                                                LXI     D,TEMPB_DAT     ; PATTERN INTO TEMPB_DAT.
78D4 1953 0000'CD                                                 CALL    MOVER_4
78D7 1954
78D7 1955 C5 46 00'00'21          ******************              DO      32
          77 20 3E
78DF 1956                         *
78DF 1957 00'8B'21                *                               LXI     H,TEMPB_DAT     ;LOAD TEST PATTERN INTO EXP_DAT.
78E2 1958 00'90'11                *                               LXI     D,EXP_DAT
78E5 1959 0000'CD                 *                               CALL    MOVER_4
78E8 1960                         *
78E8 1961 00'8B'21                *                       3$:     LXI     H,TEMPB_DAT     ;LOAD TEST PATTERN INTO DATA0.
78EB 1962 00'00'11                *                               LXI     D,DATA0
78EE 1963 0000'CD                 *                               CALL    MOVER_4
78F1 1964                         *
78F1 1965 0205'CD                 *                               CALL    WRITE_WR1_32    ;WRITE PATTERN TO WR[1].
78F4 1966                         *
78F4 1967 023E'CD                 *                               CALL    READ_WR1_32     ;READ DATA BACK FROM WR[1].
78F7 1968                         *
78F7 1969 00'00'21                *                               LXI     H,DATA0         ;MOVE RECEIVED DATA TO REC_DAT.
78FA 1970 00'95'11                *                               LXI     D,REC_DAT
78FD 1971 0000'CD                 *                               CALL    MOVER_4
7900 1972                         *
7900 1973 00'90'21                * ****************              IFNEQ   EXP_DAT,REC_DAT,4       ;COMPARE THE EXPECTED AND
          04 0E 00'95'11          * *
          0911'CA 0000'CD         * *
790E 1974                         * *
790E 1975                         * *                                                     ; RECEIVED DATA.
790E 1976 03E7'CD                 * *                             CALL    TESTC_ERROR     ;PRINT AN ERROR MESSAGE.
7911 1977                         * *
7911 1978                         * ****************              ENDIF
7911 1979                         *
7911 1980 0000'3A                 *                               LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7914 1981 0F                      *                               RRC
7915 1982 08E8'DA                 *                               JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7918 1983                         *
7918 1984 B7                      *                               ORA     A               ;CLEAR THE CARRY.
7919 1985                         *
7919 1986 00'8A'21                *                               LXI     H,TEMPB_SHIFT   ;SHIFT THE ONE THROUGH THE ZERO FIELD.
791C 1987 0000'22                 *                               SHLD    SHIFT_PNT
791F 1988 0000'CD                 *                               CALL    SHIFTER
7922 1989                         *
7922 1990 35 00'00'21             ******************              ENDDO
          70 C1 08DF'C2
792B 1991
792B 1992
792B 1993
792B 1994 00'5A'21                                                LXI     H,ZERO_SHIFT    ;LOAD ZERO THROUGH A ONE FIELD
792E 1995 00'8B'11                                                LXI     D,TEMPB_DAT     ; PATTERN INTO TEMPB_DAT.
7931 1996 0000'CD                                                 CALL    MOVER_4
7934 1997
7934 1998 C5 46 00'00'21          ******************              DO      32
          77 20 3E                *
793C 1999                         *
793C 2000 00'8B'21                *                               LXI     H,TEMPB_DAT     ;LOAD TEST PATTERN INTO EXP_DAT.
793F 2001 00'90'11                *                               LXI     D,EXP_DAT
7942 2002 0000'CD                 *                               CALL    MOVER_4
7945 2003                         *
7945 2004 00'8B'21                *                       4$:     LXI     H,TEMPB_DAT     ;LOAD TEST PATTERN INTO DATA0.
7948 2005 00'00'11                *                               LXI     D,DATA0
794B 2006 0000'CD                 *                               CALL    MOVER_4
```

```
794E 2007                         *
794E 2008 0205'CD                 *                              CALL    WRITE_WR1_32    ;WRITE PATTERN TO WR[1].
7951 2009                         *
7951 2010 023E'CD                 *                              CALL    READ_WR1_32     ;READ DATA BACK FROM WR[1].
7954 2011                         *
7954 2012 00'00'21                *                              LXI     H,DATA0         ;PUT RECEIVED DATA INTO REC_DAT.
7957 2013 00'95'11                *                              LXI     D,REC_DAT
795A 2014 0000'CD                 *                              CALL    MOVER_4
795D 2015                         *
795D 2016 00'90'21                * ****************             IFNEQ   EXP_DAT,REC_DAT,4        ;COMPARE THE EXPECTED AND
          04 0E 00'95'11          * *
          096E'CA 0000'CD         * *
796B 2017                         * *                                                     ; RECEIVED DATA.
796B 2018                         * *
796B 2019 03E7'CD                 * *                            CALL    TESTC_ERROR     ;PRINT AN ERROR MESSAGE.
796E 2020                         * *
796E 2021                         * ****************             ENDIF
796E 2022                         *
796E 2023 0000'3A                 *                              LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7971 2024 0F                      *                              RRC
7972 2025 0945'DA                 *                              JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7975 2026                         *
7975 2027 37                      *                              STC                     ;SET THE CARRY.
7976 2028                         *
7976 2029 00'8A'21                *                              LXI     H,TEMPB_SHIFT   ;SHIFT THE ZERO THROUGH THE ONE FIELD.
7979 2030 0000'22                 *                              SHLD    SHIFT_PNT
797C 2031 0000'CD                 *                              CALL    SHIFTER
797F 2032                         *
797F 2033 35 00'00'21             ******************             ENDDO
          70 C1 093C'C2
7988 2034
7988 2035
7988 2036 02 3E                                                  MVI     A,2             ;SECOND PART OF TEST C.
798A 2037 0000'32                                                STA     CON_ERR_NUM
798D 2038
798D 2039 00'59'21                                    5$:        LXI     H,ONE_DAT       ;LOAD ALL ONES INTO DATA0.
7990 2040 00'00'11                                               LXI     D,DATA0
7993 2041 0000'CD                                                CALL    MOVER_4
7996 2042
7996 2043 0172'CD                                                CALL    FAST_WRITE_WR0  ;WRITE ONES TO WR0.
7999 2044
7999 2045 00'54'21                                               LXI     H,ZERO_DAT      ;LOAD ALL ZEROS INTO DATA0.
799C 2046 00'00'11                                               LXI     D,DATA0
799F 2047 0000'CD                                                CALL    MOVER_4
79A2 2048
79A2 2049 0205'CD                                                CALL    WRITE_WR1_32    ;WRITE ZEROS TO WR1.
79A5 2050
79A5 2051 01B9'CD                                                CALL    FAST_READ_WR0   ;READ WR0.
79A8 2052
79A8 2053 00'59'21                                               LXI     H,ONE_DAT       ;MOVE ALL ONES INTO EXP_DAT.
79AB 2054 00'90'11                                               LXI     D,EXP_DAT
79AE 2055 0000'CD                                                CALL    MOVER_4
79B1 2056
79B1 2057 00'00'21                                               LXI     H,DATA0         ;MOVE RECEIVED DATA TO REC_DAT.
79B4 2058 00'95'11                                               LXI     D,REC_DAT
79B7 2059 0000'CD                                                CALL    MOVER_4
79BA 2060
79BA 2061 00'90'21                ******************             IFNEQ   EXP_DAT,REC_DAT,4        ;COMPARE THE EXPECTED AND
          04 0E 00'95'11          *
          09CB'CA 0000'CD         *
```

```
79C8 2062                   *                                                        ; RECEIVED DATA.
79C8 2063                   *
79C8 2064 03E7'CD           *                       CALL    TESTC_ERROR     ;PRINT AN ERROR MESSAGE.
79CB 2065                   *
79CB 2066                   ******************      ENDIF
79CB 2067
79CB 2068 0000'3A                                   LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
79CE 2069 0F                                        RRC
79CF 2070 098D'DA                                   JC      5$              ;...JUMP TO BEGINNING OF ERROR LOOP.
79D2 2071
79D2 2072 00'59'21                          6$:     LXI     H,ONE_DAT       ;LOAD ALL ONES INTO DATA0.
79D5 2073 00'00'11                                  LXI     D,DATA0
79D8 2074 0000'CD                                   CALL    MOVER_4
79DB 2075
79DB 2076 0172'CD                                   CALL    FAST_WRITE_WR0  ;WRITE ONES TO WR0.
79DE 2077
79DE 2078 00'54'21                                  LXI     H,ZERO_DAT      ;LOAD ALL ZEROS INTO DATA0.
79E1 2079 00'00'11                                  LXI     D,DATA0
79E4 2080 0000'CD                                   CALL    MOVER_4
79E7 2081
79E7 2082 0205'CD                                   CALL    WRITE_WR1_32    ;WRITE ZEROS TO WR1.
79EA 2083
79EA 2084 023E'CD                                   CALL    READ_WR1_32     ;READ WR1.
79ED 2085
79ED 2086 00'54'21                                  LXI     H,ZERO_DAT      ;MOVE ALL ZEROS INTO EXP_DAT.
79F0 2087 00'90'11                                  LXI     D,EXP_DAT
79F3 2088 0000'CD                                   CALL    MOVER_4
79F6 2089
79F6 2090 00'00'21                                  LXI     H,DATA0         ;MOVE READ DATA INTO REC_DAT.
79F9 2091 00'95'11                                  LXI     D,REC_DAT
79FC 2092 0000'CD                                   CALL    MOVER_4
79FF 2093
79FF 2094 00'90'21          ******************      IFNEQ   EXP_DAT,REC_DAT,4       ;COMPARE THE EXPECTED AND
          04 0E 00'95'11    *
          0A10'CA 0000'CD   *
7A0D 2095                   *                                                       ; RECEIVED DATA.
7A0D 2096                   *
7A0D 2097 03E7'CD           *                       CALL    TESTC_ERROR     ;PRINT AN ERROR MESSAGE.
7A10 2098                   *
7A10 2099                   ******************      ENDIF
7A10 2100
7A10 2101 0000'3A                                   LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7A13 2102 0F                                        RRC
7A14 2103 09D2'DA                                   JC      6$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7A17 2104
7A17 2105 3C D3                                     TAIL
7A19 2106
7A19 2107
7A19 2108                           ;*********************************************************************************
7A19 2109                           ;
7A19 2110                           ;
7A19 2111                           ;       TESTD           32 BIT D AND Y BUSES TEST
7A19 2112                           ;       -----           -------------------------
7A19 2113                           ;
7A19 2114                           ;               THE REMAINING 24 BITS OF THE D AND Y BUSES THAT WHERE NOT
7A19 2115                           ;       TESTED BY THE 8 BIT LOADS WILL NOW BE TESTED.  32 BITS OF DATA WILL
7A19 2116                           ;       BE LOADED INTO A 2901 WORKING REGISTER 8 BITS AT A TIME. THE DATA
7A19 2117                           ;       WILL BE WRITTEN TO AN LS LOCATION AND THEN READ BACK.  THE DATA USED
7A19 2118                           ;       WILL BE THE STANDARD PATTERN OF A ONE RIPPLED THROUGH A FIELD OF
7A19 2119                           ;       ZEROS AND THEN A ZERO RIPPLED THROUGH A FIELD OF ONES.
```

```
7A19 2120                                 ;
7A19 2121                                 ;
7A19 2122                                 ;       TEST STEPS:
7A19 2123                                 ;
7A19 2124                                 ;               1) LOAD WR WITH A DATA PATTERN.
7A19 2125                                 ;               2) WRITE THE PATTERN TO LS (MOV WR TO LS INST IN CSR).
7A19 2126                                 ;               3) CLEAR THE WORKING REGISTER.
7A19 2127                                 ;               4) READ THE DATA PATTERN BACK TO WR FROM LS (MOV LS TO WR).
7A19 2128                                 ;               5) READ WR BACK TO THE 8085 8 BITS AT A TIME.
7A19 2129                                 ;               6) PRINT ERROR IF ANY.
7A19 2130                                 ;               7) REPEAT 1-6 FOR EACH DATA PATTERN.
7A19 2131                                 ;
7A19 2132                                 ;
7A19 2133                                 ;       ASSUMPTIONS:  THE CLEAR WR INSTRUCTION FROM THE PREVIOUS TEST IS
7A19 2134                                 ;                     WORKING AND THAT THE WR IS WORKING ALSO FROM PREVIOUS
7A19 2135                                 ;                     TESTING.
7A19 2136                                 ;
7A19 2137                                 ;
7A19 2138                                 ;       ERROR1: 32 BIT D AND Y BUS TEST FAILURE
7A19 2139                                 ;
7A19 2140                                 ;       DEBUG:  1) CHECK THE Y BUS TO SEE THAT THE DATA REACHES LS
7A19 2141                                 ;                  FROM THE 2901.
7A19 2142                                 ;               2) CHECK LS TO SEE THAT THE LOCATION USED(0) IS CORRECT
7A19 2143                                 ;                  ON ''LS ADRS L''.
7A19 2144                                 ;               3) CHECK ''LS WRT EN L'' AND ''ENABLE LONG H''.
7A19 2145                                 ;               4) CHECK THE D BUS TO SEE THAT THE DATA RETURNING FROM LS
7A19 2146                                 ;                  IS CORRECT.
7A19 2147                                 ;               5) CHECK ''ENABLE LS HI L'' FOR LOW ON THE READ.
7A19 2148                                 ;               6) REMOVE OTHER MODULES ON THE Y-BUS IF THE BUS APPEARS
7A19 2149                                 ;                  TO BE HELD DOWN.
7A19 2150                                 ;
7A19 2151                                 ;
7A19 2152                                 ;
7A19 2153                                 ;*******************************************************************************
7A19 2154
7A19 2155
7A19 2156
7A19 2157 0000'CD 0D 3E                    TESTD:          HEAD    <^X0D>
          0F 0000'3A
          C9 0A26'D2
          0F 0000'3A
          3D D3 0BAB'DA
7A2F 2158
7A2F 2159 00B9'CD                                         CALL    LD_SUPPORT      ;LOAD SUPPORT CODE FOR FAST READS AND
7A32 2160                                                                         ; WRITES OF WR0.
7A32 2161
7A32 2162 00'54'21                                        LXI     H,ZERO_DAT      ;LOAD ALL ZEROS INTO EXP_DAT.
7A35 2163 00'90'11                                        LXI     D,EXP_DAT
7A38 2164 0000'CD                                         CALL    MOVER_4
7A3B 2165
7A3B 2166 00'54'21                         1$:            LXI     H,ZERO_DAT      ;LOAD ALL ZEROS DATA0.
7A3E 2167 00'00'11                                        LXI     D,DATA0
7A41 2168 0000'CD                                         CALL    MOVER_4
7A44 2169
7A44 2170 0172'CD                                         CALL    FAST_WRITE_WR0  ;WRITE THE PATTERN TO WR0.
7A47 2171
7A47 2172 00'11'21                                        LXI     H,MOV_WR0_LS0   ;MOVE WR0 TO LS0.
7A4A 2173 0294'CD                                         CALL    EXEC_INST
7A4D 2174
7A4D 2175 00'01'21                                        LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
```

```
7A50 2176 0294'CD                                               CALL    EXEC_INST
7A53 2177
7A53 2178 00'14'21                                              LXI     H,MOV_LSO_WRO   ;MOVE LSO TO WRO.
7A56 2179 0294'CD                                               CALL    EXEC_INST
7A59 2180
7A59 2181 01B9'CD                                               CALL    FAST_READ_WRO   ;READ WRO.
7A5C 2182
7A5C 2183 00'00'21                                              LXI     H,DATA0         ;MOVE THE RECEIVED DATA TO REC_DAT.
7A5F 2184 00'95'11                                              LXI     D,REC_DAT
7A62 2185 0000'CD                                               CALL    MOVER_4
7A65 2186
7A65 2187 00'90'21             ******************               IFNEQ   EXP_DAT,REC_DAT,4    ;COMPARE THE EXPECTED AND
          04 0E 00'95'11       *
          0A76'CA 0000'CD      *
7A73 2188                      *                                                             ; RECEIVED DATA.
7A73 2189                      *
7A73 2190 03E2'CD              *                                CALL    TESTD_ERROR     ;PRINT AN ERROR MESSAGE.
7A76 2191                      *
7A76 2192                      ******************               ENDIF
7A76 2193
7A76 2194 0000'3A                                               LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7A79 2195 0F                                                    RRC
7A7A 2196 0A3B'DA                                               JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7A7D 2197
7A7D 2198
7A7D 2199 00'59'21                                              LXI     H,ONE_DAT       ;LOAD ALL ONES INTO EXP_DAT.
7A80 2200 00'90'11                                              LXI     D,EXP_DAT
7A83 2201 0000'CD                                               CALL    MOVER_4
7A86 2202
7A86 2203 00'59'21                                2$:           LXI     H,ONE_DAT       ;LOAD THE PATTERN INTO DATA0.
7A89 2204 00'00'11                                              LXI     D,DATA0
7A8C 2205 0000'CD                                               CALL    MOVER_4
7A8F 2206
7A8F 2207 0172'CD                                               CALL    FAST_WRITE_WRO  ;WRITE THE PATTERN TO WRO.
7A92 2208
7A92 2209 00'11'21                                              LXI     H,MOV_WRO_LSO   ;MOVE WRO TO LSO.
7A95 2210 0294'CD                                               CALL    EXEC_INST
7A98 2211
7A98 2212 00'01'21                                              LXI     H,X_CLR_WRO+1   ;CLEAR WRO.
7A9B 2213 0294'CD                                               CALL    EXEC_INST
7A9E 2214
7A9E 2215 00'14'21                                              LXI     H,MOV_LSO_WRO   ;MOVE LSO TO WRO.
7AA1 2216 0294'CD                                               CALL    EXEC_INST
7AA4 2217
7AA4 2218 01B9'CD                                               CALL    FAST_READ_WRO   ;READ WRO.
7AA7 2219
7AA7 2220 00'00'21                                              LXI     H,DATA0         ;MOVE THE RECEIVED DATA TO REC_DAT.
7AAA 2221 00'95'11                                              LXI     D,REC_DAT
7AAD 2222 0000'CD                                               CALL    MOVER_4
7AB0 2223
7AB0 2224 00'90'21             ******************               IFNEQ   EXP_DAT,REC_DAT,4    ;COMPARE THE EXPECTED AND
          04 0E 00'95'11       *
          0AC1'CA 0000'CD      *
7ABE 2225                      *                                                             ; RECEIVED DATA.
7ABE 2226                      *
7ABE 2227 03E2'CD              *                                CALL    TESTD_ERROR     ;PRINT AN ERROR MESSAGE.
7AC1 2228                      *
7AC1 2229                      ******************               ENDIF
7AC1 2230
7AC1 2231 0000'3A                                               LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
```

```
7AC4 2232 0F                                                    RRC
7AC5 2233 0AB6'DA                                               JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7AC8 2234
7AC8 2235
7AC8 2236 00'55'21                                              LXI     H,ONE_SHIFT     ;LOAD A ONE THROUGH A ZERO FIELD
7ACB 2237 00'8B'11                                              LXI     D,TEMPB_DAT     ; PATTERN INTO TEMPB_DAT.
7ACE 2238 0000'CD                                               CALL    MOVER_4
7AD1 2239
7AD1 2240 C5 46 00'00'21        ******************              DO      32
          77 20 3E              *
7AD9 2241                       *
7AD9 2242 00'8B'21              *                               LXI     H,TEMPB_DAT     ;LOAD THE PATTERN INTO EXP_DAT.
7ADC 2243 00'90'11              *                               LXI     D,EXP_DAT
7ADF 2244 0000'CD               *                               CALL    MOVER_4
7AE2 2245                       *
7AE2 2246 00'8B'21              *                       3$:     LXI     H,TEMPB_DAT     ;LOAD THE PATTERN INTO DATA0.
7AE5 2247 00'00'11              *                               LXI     D,DATA0
7AE8 2248 0000'CD               *                               CALL    MOVER_4
7AEB 2249                       *
7AEB 2250 0172'CD               *                               CALL    FAST_WRITE_WR0  ;WRITE THE PATTERN TO WR0.
7AEE 2251                       *
7AEE 2252 00'11'21              *                               LXI     H,MOV_WR0_LS0   ;MOVE WR0 TO LS0.
7AF1 2253 0294'CD               *                               CALL    EXEC_INST
7AF4 2254                       *
7AF4 2255 00'01'21              *                               LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
7AF7 2256 0294'CD               *                               CALL    EXEC_INST
7AFA 2257                       *
7AFA 2258 00'14'21              *                               LXI     H,MOV_LS0_WR0   ;MOVE LS0 TO WR0.
7AFD 2259 0294'CD               *                               CALL    EXEC_INST
7B00 2260                       *
7B00 2261 01B9'CD               *                               CALL    FAST_READ_WR0   ;READ WR0.
7B03 2262                       *
7B03 2263 00'00'21              *                               LXI     H,DATA0         ;MOVE THE RECEIVED DATA TO REC_DAT.
7B06 2264 00'95'11              *                               LXI     D,REC_DAT
7B09 2265 0000'CD               *                               CALL    MOVER_4
7B0C 2266                       *
7B0C 2267 00'90'21              * ****************              IFNEQ   EXP_DAT,REC_DAT,4     ;COMPARE THE EXPECTED AND
          04 0E 00'95'11        * *
          0B1D'CA 0000'CD       * *
7B1A 2268                       * *                                                           ; RECEIVED DATA.
7B1A 2269                       * *
7B1A 2270 03E2'CD               * *                             CALL    TESTD_ERROR     ;PRINT AN ERROR MESSAGE.
7B1D 2271                       * *
7B1D 2272                       * ****************              ENDIF
7B1D 2273                       *
7B1D 2274 0000'3A               *                               LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7B20 2275 0F                    *                               RRC
7B21 2276 0AE2'DA               *                               JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7B24 2277                       *
7B24 2278 B7                    *                               ORA     A               ;CLEAR THE CARRY.
7B25 2279                       *
7B25 2280 00'8A'21              *                               LXI     H,TEMPB_SHIFT   ;SHIFT THE ONE THROUGH THE ZERO FIELD.
7B28 2281 0000'22               *                               SHLD    SHIFT_PNT
7B2B 2282 0000'CD               *                               CALL    SHIFTER
7B2E 2283                       *
7B2E 2284 35 00'00'21           ******************              ENDDO
          70 C1 0AD9'C2
7B37 2285
7B37 2286 00A9'CD                                               CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7B3A 2287
```

```
7B3A 2288
7B3A 2289 00'5A'21                                             LXI     H,ZERO_SHIFT    ;LOAD A ZERO THROUGH A ONE FIELD
7B3D 2290 00'8B'11                                             LXI     D,TEMPB_DAT     ; PATTERN INTO TEMPB_DAT.
7B40 2291 0000'CD                                              CALL    MOVER_4
7B43 2292
7B43 2293 C5 46 00'00'21       *******************             DO      32
          77 20 3E             *
7B4B 2294                      *
7B4B 2295 00'8B'21             *                               LXI     H,TEMPB_DAT     ;LOAD THE PATTERN INTO EXP_DAT.
7B4E 2296 00'90'11             *                               LXI     D,EXP_DAT
7B51 2297 0000'CD              *                               CALL    MOVER_4
7B54 2298                      *
7B54 2299 00'8B'21             *                  4$:          LXI     H,TEMPB_DAT     ;LOAD THE PATTERN INTO DATA0.
7B57 2300 00'00'11             *                               LXI     D,DATA0
7B5A 2301 0000'CD              *                               CALL    MOVER_4
7B5D 2302                      *
7B5D 2303 0172'CD              *                               CALL    FAST_WRITE_WR0  ;WRITE THE PATTERN TO WR0.
7B60 2304                      *
7B60 2305 00'11'21             *                               LXI     H,MOV_WR0_LS0   ;MOVE WR0 TO LS0.
7B63 2306 0294'CD              *                               CALL    EXEC_INST
7B66 2307                      *
7B66 2308 00'01'21             *                               LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
7B69 2309 0294'CD              *                               CALL    EXEC_INST
7B6C 2310                      *
7B6C 2311 00'14'21             *                               LXI     H,MOV_LS0_WR0   ;MOVE LS0 TO WR0.
7B6F 2312 0294'CD              *                               CALL    EXEC_INST
7B72 2313                      *
7B72 2314 01B9'CD              *                               CALL    FAST_READ_WR0   ;READ WR0.
7B75 2315                      *
7B75 2316 00'00'21             *                               LXI     H,DATA0         ;MOVE THE RECEIVED DATA TO REC_DAT.
7B78 2317 00'95'11             *                               LXI     D,REC_DAT
7B7B 2318 0000'CD              *                               CALL    MOVER_4
7B7E 2319                      *
7B7E 2320 00'90'21             * *****************             IFNEQ   EXP_DAT,REC_DAT,4    ;COMPARE THE EXPECTED AND
          04 0E 00'95'11       * *
          0B8F'CA 0000'CD      * *
7B8C 2321                      * *                                                           ; RECEIVED DATA.
7B8C 2322                      * *
7B8C 2323 03E2'CD              * *                             CALL    TESTD_ERROR     ;PRINT AN ERROR MESSAGE.
7B8F 2324                      * *
7B8F 2325                      * *****************             ENDIF
7B8F 2326                      *
7B8F 2327 0000'3A              *                               LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7B92 2328 0F                   *                               RRC
7B93 2329 0B54'DA              *                               JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7B96 2330                      *
7B96 2331 37                   *                               STC                     ;SET THE CARRY.
7B97 2332                      *
7B97 2333 00'8A'21             *                               LXI     H,TEMPB_SHIFT   ;SHIFT THE ZERO THROUGH THE ONE FIELD.
7B9A 2334 0000'22              *                               SHLD    SHIFT_PNT
7B9D 2335 0000'CD              *                               CALL    SHIFTER
7BA0 2336                      *
7BA0 2337 35 00'00'21          *******************             ENDDO
          70 C1 0B4B'C2
7BA9 2338
7BA9 2339 3C D3                                                TAIL
7BAB 2340
7BAB 2341
7BAB 2342
7BAB 2343                                         ;*******************************************************************************
```

```
                              ;
                              ;
                              ;       TESTE           LS RAM DATA TEST
                              ;       -----           ----------------
                              ;
                              ;               THE LS RAM DATA TEST TESTS EACH LOCATION OF THE RAM
                              ;       SO THAT EACH 4 BIT SLICE GETS TESTED WITH RIPPLED 0 THROUGH 1'S AND
                              ;       ONE RIPPLED THROUGH ZEROS. THE LOCATIONS TO BE TESTED ARE FROM
                              ;       0 TO F7 EXCLUDING 78 - 7F. THIS IS DUE TO THE FACT THE UNTESTED
                              ;       LOCATIONS HAVE SPECIAL FUNCTIONS AND DO NOT ACTUALLY USE THE LS
                              ;       RAM (F8 - FF ARE ALSO IN THE CATEGORY).
                              ;
                              ;
                              ;       DATA PATTERNS:  FFFFFFFF
                              ;                       EEEEEEEE
                              ;                       DDDDDDDD
                              ;                       BBBBBBBB
                              ;                       77777777
                              ;                       00000000
                              ;                       11111111
                              ;                       22222222
                              ;                       44444444
                              ;                       88888888
                              ;
                              ;
                              ;       TEST STEPS:
                              ;
                              ;               1) WRITE DATA PATTERN TO WR.
                              ;               2) MOVE DATA FROM WR TO LS.
                              ;               3) CLEAR WR.
                              ;               4) READ DATA FROM LS TO WR.
                              ;               5) COMPARE DATA SENT WITH RETURNED.
                              ;               6) PRINT ERROR IF ANY.
                              ;               7) REPEAT 1 - 6 FOR ALL DATA PATTERNS.
                              ;               9) REPEAT 1 - 7 FOR ALL LS LOCATONS.
                              ;
                              ;
                              ;       ASSUMPTIONS:  THE CLEAR WR INSTRUCTION FROM THE PREVIOUS TEST IS
                              ;                     WORKING, THAT THE WR IS WORKING ALSO FROM PREVIOUS
                              ;                     TESTING, THAT THE DATA PATH TO LS IS WORKING FROM THE
                              ;                     PREVIOUS TEST.
                              ;
                              ;
                              ;       ERROR1: LS RAM DATA TEST FAILURE.
                              ;
                              ;       NOTE: ADDITIONAL DATA CONTAINS THE PRESENT LS LOCATION.
                              ;
                              ;
                              ;       DEBUG:  1) CHECK RAM MEMORY FOR ERRORS.
                              ;               2) ALTHOUGH EACH ADDRESS IN THE LS IS ACCESSED IN MOST CASES IF
                              ;                  THERE WHERE AN ADDRESS FAILURE IT WOULD GO UNDETECTED BY
                              ;                  THIS TEST.  HOWEVER IF AN ADDRESS SHOULD END UP IN THE
                              ;                  RANGE THAT CAUSES SPECIAL FUNCTIONS SUCH AN ERROR COULD
                              ;                  OCCUR, SO ''LS ADRS 0 - 7 H'' SHOULD BE CHECKED.
                              ;
                              ;
                              ;*******************************************************************************
                              
                              
0000'CD 0E 3E                 TESTE:          HEAD    <^X0E>
```

```
          0F 0000'3A
          C9 0BB5'D2
          0F 0000'3A
          3D D3 0C5E'DA
7BC1 2404
7BC1 2405 00B9'CD                                            CALL   LD_SUPPORT      ;LOAD SUPPORT FOR FAST READS AND
7BC4 2406                                                                           ; WRITES OF WRO.
7BC4 2407
7BC4 2408
7BC4 2409 01 3E                                              MVI    A,1             ;FIRST (AND ONLY) PART OF TEST E.
7BC6 2410 0000'32                                            STA    CON_ERR_NUM
7BC9 2411
7BC9 2412 00 3E                                              MVI    A,0             ;LS_ADDR POINTS TO THE LS LOACTION
7BCB 2413 0009'32                                            STA    LS_ADDR         ; BEING WRITTEN AND READ.
7BCE 2414
7BCE 2415 14 3E                                              MVI    A,20            ;INITIALIZE LOOP_CNT.
7BD0 2416 000C'32                                            STA    LOOP_CNT
7BD3 2417
7BD3 2418 00'5E'21                            1$:            LXI    H,TESTE_DAT     ;POINT AT THE FIRST OF 10 TEST DATA
7BD6 2419 000A'22                                            SHLD   DATA_PNT        ; PATTERNS.
7BD9 2420
7BD9 2421 C5 46 00'00'21    ******************               DO     10              ;TEN PATTERNS ARE WRITTEN TO LS.
          77 0A 3E          *
7BE1 2422                   *
7BE1 2423 000A'2A           *                                LHLD   DATA_PNT        ;MOVE A TEST PATTERN TO EXP_DAT.
7BE4 2424 00'90'11          *                                LXI    D,EXP_DAT
7BE7 2425 0000'CD           *                                CALL   MOVER_4
7BEA 2426                   *
7BEA 2427 000A'2A           *                 2$:            LHLD   DATA_PNT        ;MOVE A PATTERN TO DATA0.
7BED 2428 00'00'11          *                                LXI    D,DATA0
7BF0 2429 0000'CD           *                                CALL   MOVER_4
7BF3 2430                   *
7BF3 2431 0009'3A           *                                LDA    LS_ADDR         ;MOVE LS POINTER TO ACCUMULATOR.
7BF6 2432 0138'CD           *                                CALL   LS_WRITE        ;WRITE TO LS.
7BF9 2433                   *
7BF9 2434 0009'3A           *                                LDA    LS_ADDR         ;MOVE LS POINTER TO ACCUMULATOR.
7BFC 2435 0155'CD           *                                CALL   LS_READ         ;READ FROM LS.
7BFF 2436                   *
7BFF 2437 00'90'21          * ****************               IFNEQ  EXP_DAT,DATA0,4        ;COMPARE THE EXPECTED AND
          04 0E 00'00'11    * *
          0C10'CA 0000'CD   * *
7C0D 2438                   * *                                                            ; RECEIVED DATA.
7C0D 2439                   * *
7C0D 2440 040D'CD           * *                              CALL   TESTE_F_ERROR   ;PRINT AN ERROR MESSAGE.
7C10 2441                   * *
7C10 2442                   * ****************               ENDIF
7C10 2443                   *
7C10 2444 0000'3A           *                                LDA    ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7C13 2445 0F                *                                RRC
7C14 2446 0BEA'DA           *                                JC     2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7C17 2447                   *
7C17 2448 000A'2A           *                                LHLD   DATA_PNT        ;POINT AT THE NEXT DATA PATTERN.
7C1A 2449 23                *                                INX    H
7C1B 2450 23                *                                INX    H
7C1C 2451 23                *                                INX    H
7C1D 2452 23                *                                INX    H
7C1E 2453 000A'22           *                                SHLD   DATA_PNT
7C21 2454                   *
7C21 2455 35 00'00'21       ******************               ENDDO
          70 C1 0BE1'C2
```

```
7C2A 2456
7C2A 2457 00'0C'21                          LXI     H,LOOP_CNT      ;POINT TO LOOP_CNT.
7C2D 2458 35                                DCR     M               ;DECREMENT LOOP_CNT.
7C2E 2459 0C40'C2                           JNZ     3$              ;IF NOT ZERO, DON'T CHECK FOR ^C.
7C31 2460
7C31 2461 14 3E                             MVI     A,20            ;INITIALIZE LOOP_CNT.
7C33 2462 000C'32                           STA     LOOP_CNT
7C36 2463
7C36 2464 0000'CD                           CALL    GCVECT          ;IF CONTROL C HAS BEEN TYPED...
7C39 2465 0C00'3A                           LDA     CNTLC_TYPED
7C3C 2466 0F                                RRC
7C3D 2467 0099'DC                           CC      CON_SET_FOR_CO  ;...GO TO MONITOR.
7C40 2468
7C40 2469 0009'3A                   3$:     LDA     LS_ADDR         ;HAS ALL LS BEEN TESTED.
7C43 2470 F7 FE                             CPI     <^XF7>
7C45 2471 0C5C'CA                           JZ      5$
7C48 2472
7C48 2473 78 FE                             CPI     <^X78>          ;SKIP LS 70-7F AS THEY ARE USED
7C4A 2474 0C54'CA                           JZ      4$              ; FOR SPECIAL PURPOSES.
7C4D 2475
7C4D 2476 3C                                INR     A               ;INCPEMENT THE LS_POINTER.
7C4E 2477 0009'32                           STA     LS_ADDR
7C51 2478
7C51 2479 0BD3'C3                           JMP     1$
7C54 2480
7C54 2481 80 3E                     4$:     MVI     A,<^X80>        ;MOVE HEX 80 TO THE LS POINTER AND
7C56 2482 0009'32                           STA     LS_ADDR         ; RESUME TESTING.
7C59 2483
7C59 2484 0BD3'C3                           JMP     1$
7C5C 2485
7C5C 2486 3C D3                     5$:     TAIL
7C5E 2487
7C5E 2488
7C5E 2489                           ;*******************************************************************************
7C5E 2490                           ;
7C5E 2491                           ;
7C5E 2492                           ;       TESTF           LS RAM MARCH PATTERN TEST
7C5E 2493                           ;       -----           -------------------------
7C5E 2494                           ;
7C5E 2495                           ;               THE RAM MARCH PATTERN TEST WRITES A BACKGROUND OF ZEROS
7C5E 2496                           ;       THROUGH THE MEMORY THEN GOES TO EACH LOCATION WORKING ITS WAY UP
7C5E 2497                           ;       READING THE 0 BACKGROUND, WRITING A 1 AND READING THE ONE BACK.
7C5E 2498                           ;       THIS THEN LEAVES 1'S IN THE BACKGROUND AND THE TEST WILL START
7C5E 2499                           ;       AT THE HIGH ADDRESS AND READ 1'S, WRITE 0'S, READ 0'S.  THE MEMORY
7C5E 2500                           ;       WILL BE TESTED FROM 0 TO F7 EXCLUDING 78 - 7F.  THIS IS DUE TO THE
7C5E 2501                           ;       FACT THE UNTESTED LOCATIONS HAVE SPECIAL FUNCTIONS AND DO NOT
7C5E 2502                           ;       ACTUALLY USE THE LS RAM (F8 - FF ARE ALSO IN THE CATEGORY)
7C5E 2503                           ;
7C5E 2504                           ;
7C5E 2505                           ;
7C5E 2506                           ;       TEST STEPS:
7C5E 2507                           ;
7C5E 2508                           ;               1)  WRITE BACKGROUND OF ZEROS.
7C5E 2509                           ;               2)  READ AND COMPARE TO ZERO.
7C5E 2510                           ;               3)  PRINT ERROR IF ANY.
7C5E 2511                           ;               4)  WRITE 1, READ, AND COMPARE TO 1.
7C5E 2512                           ;               5)  PRINT ERROR IF ANY.
7C5E 2513                           ;               6)  REPEAT 2-5 FOR ALL LOCATIONS (LOW TO HIGH).
7C5E 2514                           ;               7)  READ AND COMPARE TO ONE.
7C5E 2515                           ;               8)  PRINT ERROR IF ANY.
```

```
7C5E 2516                                        ;               9)  WRITE 0, READ, AND COMPARE TO 0.
7C5E 2517                                        ;               10) PRINT ERROR IF ANY.
7C5E 2518                                        ;               11) REPEAT 7-10 FOR ALL LOCATIONS (HIGH TO LOW).
7C5E 2519                                        ;
7C5E 2520                                        ;
7C5E 2521                                        ;       ASSUMPTIONS:  THE DATA PATH TO AND FROM LS IS WORKING
7C5E 2522                                        ;                     FROM PREVIOUS TESTS.
7C5E 2523                                        ;
7C5E 2524                                        ;
7C5E 2525                                        ;       ERROR1: LS RAM MARCH TEST FAILURE.
7C5E 2526                                        ;
7C5E 2527                                        ;       NOTE: ADDITIONAL DATA CONTAINS PRESENT LS LOACTION.
7C5E 2528                                        ;
7C5E 2529                                        ;
7C5E 2530                                        ;       DEBUG:  1) CHECK ADDRESS DECODE FOR LS FROM THE MUX OF CSR BITS
7C5E 2531                                        ;                  AND OS BITS.  THE CSR BITS SHOULD BE ENABLED THROUGH.
7C5E 2532                                        ;               2) CHECK ''LS ADRS 0 - 7 H'' FOR ADDRESSING ERRORS.
7C5E 2533                                        ;               3) CHECK RAM MEMORY FOR ERRORS IN INTERNAL ADDRESS LOGIC.
7C5E 2534                                        ;
7C5E 2535                                        ;
7C5E 2536                                        ;********************************************************************************
7C5E 2537
7C5E 2538
7C5E 2539 0000'CD 0F 3E                          TESTF:          HEAD    <^X0F>
          0F 0000'3A
          C9 0C6B'D2
          0F 0000'3A
          3D D3 0DC6'DA
7C74 2540
7C74 2541 00B9'CD                                                CALL    LD_SUPPORT      ;LOAD SUPPORT FOR FAST READS AND
7C77 2542                                                                                ; WRITES OF WR0.
7C77 2543
7C77 2544
7C77 2545 00 3E                                                  MVI     A,0             ;ZERO LS POINTER.
7C79 2546 0009'32                                                STA     LS_ADDR
7C7C 2547
7C7C 2548 00'54'21                               1$:             LXI     H,ZERO_DAT      ;LOAD ALL ZEROS INTO DATA0.
7C7F 2549 00'00'11                                               LXI     D,DATA0
7C82 2550 0000'CD                                                CALL    MOVER_4
7C85 2551
7C85 2552 0009'3A                                                LDA     LS_ADDR         ;MOV LS POINTER TO ACCUM.
7C88 2553 0138'CD                                                CALL    LS_WRITE        ;WRITE ALL ZEROS TO LS.
7C8B 2554
7C8B 2555 0009'3A                                                LDA     LS_ADDR         ;HAS ALL LS BEEN WRITTEN?
7C8E 2556 F7 FE                                                  CPI     <^XF7>
7C90 2557 0CA7'CA                                                JZ      3$
7C93 2558
7C93 2559 77 FE                                                  CPI     <^X77>          ;HAS LS_POINTER REACHED BLOCK
7C95 2560 0C9F'CA                                                JZ      2$              ; OF LS TO BE SKIPPED.
7C98 2561
7C98 2562 3C                                                     INR     A               ;INCREMENT LS POINTER.
7C99 2563 0009'32                                                STA     LS_ADDR
7C9C 2564 0C7C'C3                                                JMP     1$              ;WRITE TO NEXT LS LOCATION.
7C9F 2565
7C9F 2566 80 3E                                  2$:             MVI     A,<^X80>        ;MOVE HEX 80 TO LS POINTER (SKIP
7CA1 2567 0009'32                                                STA     LS_ADDR         ; LOCATIONS 78-7F).
7CA4 2568 0C7C'C3                                                JMP     1$              ;WRITE TO NEXT LS LOCATION.
7CA7 2569
7CA7 2570
7CA7 2571
```

```
7CA7 2572 0000'CD                                3$:    CALL    GCVECT          ;IF CONTROL C HAS BEEN TYPED...
7CAA 2573 0000'3A                                       LDA     CNTLC_TYPED
7CAD 2574 0F                                            RRC
7CAE 2575 0099'DC                                       CC      CON_SET_FOR_CO  ;...GO TO MONITOR.
7CB1 2576
7CB1 2577
7CB1 2578
7CB1 2579 01 3E                                         MVI     A,1             ;FIRST PART OF TEST F.
7CB3 2580 0000'32                                       STA     CON_ERR_NUM
7CB6 2581
7CB6 2582 00 3E                                         MVI     A,0             ;ZERO LS POINTER.
7CB8 2583 0009'32                                       STA     LS_ADDR
7CBB 2584
7CBB 2585 0009'3A                                4$:    LDA     LS_ADDR         ;MOVE LS POINTER TO A.
7CBE 2586 0155'CD                                       CALL    LS_READ         ;READ LS.
7CC1 2587
7CC1 2588 00'54'21          *******************         IFNEQ   ZERO_DAT,DATA0,4        ;COMPARE DATA READ WITH
          04 0E 00'00'11    *
          0CDB'CA 0000'CD   *
7CCF 2589                   *                                                           ; ALL ZEROS.
7CCF 2590                   *
7CCF 2591 00'54'21          *                           LXI     H,ZERO_DAT      ;LOAD ZEROS INTO EXP_DAT.
7CD2 2592 00'90'11          *                           LXI     D,EXP_DAT
7CD5 2593 0000'CD           *                           CALL    MOVER_4
7CD8 2594                   *
7CD8 2595 040D'CD           *                           CALL    TESTE_F_ERROR   ;PRINT AN ERROR MESSAGE.
7CDB 2596                   *
7CDB 2597                   *******************         ENDIF
7CDB 2598
7CDB 2599 0000'3A                                       LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7CDE 2600 0F                                            RRC
7CDF 2601 0CBB'DA                                       JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7CE2 2602
7CE2 2603 00'59'21                               5$:    LXI     H,ONE_DAT       ;LOAD ALL ONES INTO DATA0.
7CE5 2604 00'00'11                                      LXI     D,DATA0
7CE8 2605 0000'CD                                       CALL    MOVER_4
7CEB 2606
7CEB 2607 0009'3A                                       LDA     LS_ADDR         ;LOAD LS POINTER INTO ACCUM.
7CEE 2608 0138'CD                                       CALL    LS_WRITE        ;WRITE TO LS.
7CF1 2609
7CF1 2610 04 0E 00'00'21                                ZERO    DATA0,4         ;CLEAR DATA0.
          0D 23 77 AF
          0CF7'C2
7CFD 2611
7CFD 2612 0009'3A                                       LDA     LS_ADDR         ;LOAD LS POINTER INTO ACCUM.
7D00 2613 0155'CD                                       CALL    LS_READ         ;READ LS.
7D03 2614
7D03 2615 00'59'21          *******************         IFNEQ   ONE_DAT,DATA0,4 ;COMPARE THE EXPECTED AND
          04 0E 00'00'11    *
          0D1D'CA 0000'CD   *
7D11 2616                   *                                                           ; RECEIVED DATA.
7D11 2617                   *
7D11 2618 00'59'21          *                           LXI     H,ONE_DAT       ;LOAD ALL ONES INTO EXP_DAT.
7D14 2619 00'90'11          *                           LXI     D,EXP_DAT
7D17 2620 0000'CD           *                           CALL    MOVER_4
7D1A 2621                   *
7D1A 2622 040D'CD           *                           CALL    TESTE_F_ERROR   ;PRINT AN ERROR MESSAGE.
7D1D 2623                   *
7D1D 2624                   *******************         ENDIF
7D1D 2625
```

```
7D1D 2626 0000'3A                                          LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7D20 2627 0F                                               RRC
7D21 2628 0CE2'DA                                          JC      5$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7D24 2629
7D24 2630
7D24 2631 0009'3A                                          LDA     LS_ADDR         ;HAVE ALL LS LOCATIONS BEEN TESTED
7D27 2632 F7 FE                                            CPI     <^XF7>          ; FOR FIRST MAIN PASS?
7D29 2633 0D40'CA                                          JZ      7$
7D2C 2634
7D2C 2635 77 FE                                            CPI     <^X77>          ;HAS LS POINTER REACHED LS BLOCK
7D2E 2636 0D38'CA                                          JZ      6$              ; TO BE SKIPPED?
7D31 2637
7D31 2638 3C                                               INR     A               ;INCREMENT THE LS POINTER.
7D32 2639 0009'32                                          STA     LS_ADDR
7D35 2640 0CBB'C3                                          JMP     4$              ;TEST NEXT LS LOCATION.
7D38 2641
7D38 2642 80 3E                                    6$:     MVI     A,<^X80>        ;MOVE HEX 80 TO LS POINTER (SKIP
7D3A 2643 0009'32                                          STA     LS_ADDR         ; LOCATIONS 78-7F).
7D3D 2644 0CBB'C3                                          JMP     4$              ;TEST NEXT LS LOCATION.
7D40 2645
7D40 2646
7D40 2647
7D40 2648 0000'CD                                  7$:     CALL    GCVECT          ;IF CONTROL C HAS BEEN TYPED...
7D43 2649 0000'3A                                          LDA     CNTLC_TYPED
7D46 2650 0F                                               RRC
7D47 2651 0099'DC                                          CC      CON_SET_FOR_CO  ;...GO TO MONITOR.
7D4A 2652
7D4A 2653
7D4A 2654 02 3E                                            MVI     A,2             ;SECOND PART OF TEST F.
7D4C 2655 0000'32                                          STA     CON_ERR_NUM
7D4F 2656
7D4F 2657 F7 3E                                            MVI     A,<^XF7>        ;POINT TO TOP OF LS.
7D51 2658 0009'32                                          STA     LS_ADDR
7D54 2659
7D54 2660 0009'3A                                  8$:     LDA     LS_ADDR         ;MOVE LS POINTER TO A.
7D57 2661 0155'CD                                          CALL    LS_READ         ;READ LS.
7D5A 2662
7D5A 2663 00'59'21           *******************           IFNEQ   ONE_DAT,DATA0,4          ;COMPARE DATA READ WITH
          04 0E 00'00'11     *
          0D74'CA 0000'CD    *
7D68 2664                    *                                                           ; ALL ONES.
7D68 2665                    *
7D68 2666 00'59'21           *                             LXI     H,ONE_DAT       ;LOAD ALL ONES INTO EXP_DAT.
7D6B 2667 00'90'11           *                             LXI     D,EXP_DAT
7D6E 2668 0000'CD            *                             CALL    MOVER_4
7D71 2669                    *
7D71 2670 040D'CD            *                             CALL    TESTE_F_ERROR   ;PRINT AN ERROR MESSAGE.
7D74 2671                    *
7D74 2672                    *******************           ENDIF
7D74 2673
7D74 2674 0000'3A                                          LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7D77 2675 0F                                               RRC
7D78 2676 0D54'DA                                          JC      8$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7D7B 2677
7D7B 2678 00'54'21                                 9$:     LXI     H,ZERO_DAT      ;LOAD ALL ZEROS INTO DATA0.
7D7E 2679 00'00'11                                         LXI     D,DATA0
7D81 2680 0000'CD                                          CALL    MOVER_4
7D84 2681
7D84 2682 0009'3A                                          LDA     LS_ADDR         ;MOVE LS POINTER TO A.
7D87 2683 0138'CD                                          CALL    LS_WRITE        ;WRITE TO LS.
```

```
7D8A 2684
7D8A 2685 0009'3A                                    LDA     LS_ADDR          ;MOVE LS POINTER TO A.
7D8D 2686 0155'CD                                    CALL    LS_READ          ;READ LS.
7D90 2687
7D90 2688 00'54'21          ******************       IFNEQ   ZERO_DAT,DATA0,4         ;COMPARE THE EXPECTED AND
          04 0E 00'00'11    *
          0DA1'CA 0000'CD   *
7D9E 2689                   *                                                          ; RECEIVED DATA.
7D9E 2690                   *
7D9E 2691 040D'CD           *                        CALL    TESTE_F_ERROR    ;PRINT AN ERROR MESSAGE.
7DA1 2692                   *
7DA1 2693                   ******************       ENDIF
7DA1 2694
7DA1 2695 0000'3A                                    LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7DA4 2696 0F                                         RRC
7DA5 2697 0D7B'DA                                    JC      9$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7DA8 2698
7DA8 2699 0009'3A                                    LDA     LS_ADDR          ;HAVE ALL LS LOCATIONS BEEN TESTED?
7DAB 2700 00 FE                                      CPI     <^X00>
7DAD 2701 0DC4'CA                                    JZ      11$
7DB0 2702
7DB0 2703 80 FE                                      CPI     <^X80>           ;HAS LS POINTER REACHED LS BLOCK
7DB2 2704 0DBC'CA                                    JZ      10$              ; TO BE SKIPPED?
7DB5 2705
7DB5 2706 3D                                         DCR     A                ;DECREMENT THE LS POINTER.
7DB6 2707 0009'32                                    STA     LS_ADDR
7DB9 2708 0D54'C3                                    JMP     8$               ;TEST NEXT LS LOCATION.
7DBC 2709
7DBC 2710 77 3E                          10$:        MVI     A,<^X77>         ;MOVE HEX 80 TO LS POINTER (SKIP
7DBE 2711 0009'32                                    STA     LS_ADDR          ; LOCATIONS 78-7F).
7DC1 2712 0D54'C3                                    JMP     8$               ;TEST NEXT LS LOCATION.
7DC4 2713
7DC4 2714 3C D3                          11$:        TAIL
7DC6 2715
7DC6 2716
7DC6 2717                                ;****************************************************************************
7DC6 2718                                ;
7DC6 2719                                ;  LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
7DC6 2720                                ;
7DC6 2721                                ;****************************************************************************
7DC6 2722
7DC6 2723
7DC6 2724 0000'CD 51 3E                  END_TEST:   HEAD    <^X51>
          0F 0000'3A
          C9 0DD3'D2
          0F 0000'3A
          3D D3 0DDE'DA
7DDC 2725
7DDC 2726 3C D3                                      TAIL
7DDE 2727
```

```
$PSW                                 = 00000006              DCLO                                 = 00000002
$SP                                  = 00000006              DISMEM                               = 000000A7
A                                    = 00000007              DISPAR                               = 000000A1
ALLOWP                               = 00000003              DOLOOP                                 ********   X    02
APTFLG                               = 00000004              E                                    = 00000003
AUTO                                 = 00000005              ENBMEM                               = 000000A6
B                                    = 00000000              ENBPAR                               = 000000A0
BOOTEN                               = 00000007              END_TEST                               00000DC6 R      02
BOOTF                                = 00000001              ENTRY                                  00000000 R      02
C                                    = 00000001              EOS_FLG                                ********   X    02
CHECK_RD                               000000A9 R      02    ERROR_CON                              ********   X    02
CLRACK                               = 000000A9              EXEC_INST                              00000294 R      02
CLRACL                               = 00000033              EXP_DAT                                00000090 R      02
CLRATN                               = 000000AB              EXP_SHIFT                              0000008F R      02
CLRBSY                               = 0000002E              FAST_READ_WR0                          000001B9 R      02
CLRCLK                               = 00000020              FAST_WRITE_WR0                         00000172 R      02
CLRCSK                               = 00000024              FILE_NAME                              00000005 R      02
CLRCSR                               = 00000038              GCVECT                                 ********   X    02
CLRDCL                               = 00000031              H                                    = 00000004
CLRHLT                               = 000000AF              HEAD                                 = 00000000
CLRLIT                               = 0000003C              HLTFLG                               = 00000002
CLRMCI                               = 00000029              INC_WORD                               ********   X    02
CLRMCK                               = 0000002B              INITH                                = 00000035
CLRPFI                               = 000000AD              INITL                                = 00000034
CLRSS                                = 00000022              ITCSR                                = 0000001D
CLRTIM                               = 000000A5              ITDB                                 = 0000001C
CLRTMR                               = 0000002C              L                                    = 00000005
CLRUPC                               = 000000A8              LD_CSR_INST                            0000029C R      02
CLRUPK                               = 00000026              LD_SUPPORT                             000000B9 R      02
CLRWCK                               = 00000036              LITERAL                              = 00000001
CNT                                  = 0000081A              LOAD_MOD_DAT                           000002D2 R      02
CNTA                                 = 00000FA0              LOAD_UPC                               ********   X    02
CNTLC_TYPED                            ********   X    02    LOOP_CNT                               0000000C R      02
COMPARE                                ********   X    02    LS_ADDR                                00000009 R      02
CON_ADD_DAT                            ********   X    02    LS_READ                                00000155 R      02
CON_BEGIN_TEST                         ********   X    02    LS_WRITE                               00000138 R      02
CON_ERROR                              ********   X    02    LVL                                  = 00000000
CON_ERR_NUM                            ********   X    02    M                                    = 00000006
CON_EXP_DAT                            ********   X    02    MACSTK1                              = 0000081A
CON_MOD_DAT                            ********   X    02    MACSTK2                              = 00000818
CON_MSK_DAT                            ********   X    02    MACSTK3                              = 00000811
CON_REC_DAT                            ********   X    02    MCPU_A                                 ********   X    02
CON_SET_FOR_CO                         00000099 R      02    MICRO_STEP_CPU                         ********   X    02
CPATTN                               = 00000083              MIPFLG                               = 000040D9
CPUACK                               = 00000082              MISC2                                = 00000001
CSR07                                = 00000085              MISC2_WR_CRR                           0000002F R      02
CSR15                                = 00000086              MOVER                                  ********   X    02
CSR23                                = 00000087              MOVER_3                                ********   X    02
CSR_SHIFT                              ********   X    02    MOVER_4                                ********   X    02
CSR_VALUE                              ********   X    02    MOV_CCR_WR0                            0000000E R      02
D                                    = 00000002              MOV_LS0_WR0                            00000014 R      02
DATA0                                  ********   X    02    MOV_LSX_WR0                            0000001A R      02
DATA1                                  ********   X    02    MOV_WR0_LS0                            00000011 R      02
DATA2                                  ********   X    02    MOV_WR0_LSX                            00000017 R      02
DATA3                                  ********   X    02    MOV_WR1_WR1                            0000001D R      02
DATA_PNT                               0000000A R      02    MWCS_A                                 ********   X    02
DATA_SHIFT                             ********   X    02    NA_OTHER                               ********   X    02
```

```
NOP_CSR                          ********   X   02
OKVT5                          = 00000007
ONE_DAT                          00000059 R     02
ONE_SHIFT                        00000055 R     02
PARITY_VECTOR                  = 00004C22
POWER_VECTOR                   = 00004C25
READ                           = 00000080
READJ1                         = 00000003
READJ2                         = 00000004
READ_BYTE_WR0                    000001D9 R     02
READ_WR1_32                      0000023E R     02
REC_DAT                          00000095 R     02
REC_SHIFT                        00000094 R     02
REMDIS                         = 00000006
REPORT_CPU_ERROR                 00000446 R     02
REPORT_ERROR                     0000044C R     02
ROT_WRT_L                        00000023 R     02
SAVE_LS_ADDR                     0000000D R     02
SETACK                         = 000000A8
SETACL                         = 00000032
SETATN                         = 000000AA
SETBSY                         = 0000002F
SETCLK                         = 00000021
SETCSK                         = 00000025
SETCSR                         = 00000039
SETDCL                         = 00000030
SETHLT                         = 000000AE
SETLIT                         = 0000003D
SETMCI                         = 00000028
SETMCK                         = 0000002A
SETPFI                         = 000000AC
SETSS                          = 00000023
SETTIM                         = 000000A4
SETTMR                         = 0000002D
SETUPC                         = 000000A9
SETUPK                         = 00000027
SETWCK                         = 00000037
SET_COMPARE                      0000011B R     02
SET_FOR_CONT                     ********   X   02
SHIFTER                          ********   X   02
SHIFTER1                         ********   X   02
SHIFT_CNT                        00000007 R     02
SHIFT_CNT_DAT                    00000008 R     02
SHIFT_L_2901                     ********   X   02
SHIFT_PNT                        ********   X   02
SHIFT_WR1_L                      00000026 R     02
SHIFT_WR1_R                      00000020 R     02
SHIFT_WR1_R8                     0000027C R     02
SKIP_TEST                        ********   X   02
SUPPORT_CODE                     00000032 R     02
SYM                            = 0000081A
TEMPA_DAT                        00000087 R     02
TEMPA_SHIFT                      00000086 R     02
TEMPB_DAT                        0000008B R     02
TEMPB_SHIFT                      0000008A R     02
TEST9                            00000450 R     02
TEST9_1_0                        000002DB R     02
TEST9_2_0                        0000033F R     02
TEST9_ERR2_3                     00000380 R     02
TEST9_ERROR                      0000037B R     02
TEST9_SHIFT                      00000307 R     02
TESTA                            00000515 R     02
TESTA_ERROR                      000003A0 R     02
TESTB                            00000609 R     02
TESTB_ERROR                      000003CD R     02
TESTC                            0000083E R     02
TESTC_ERROR                      000003E7 R     02
TESTD                            00000A19 R     02
TESTD_ERROR                      000003E2 R     02
TESTE                            00000BAB R     02
TESTE_DAT                        0000005E R     02
TESTE_F_ERROR                    0000040D R     02
TESTF                            00000C5E R     02
TTCR                           = 0000004B
TTDB                           = 00000048
TTMODE                         = 0000004A
TTSR                           = 00000049
TUCR                           = 00000047
TUDB                           = 00000044
TUMODE                         = 00000046
TUSR                           = 00000045
UPC14                          = 00000084
UPC14A                         = 00000000
UPC_VALUE                        ********   X   02
VERSION                          00000003 R     02
VER_WRITE_TAB                    00000044 R     02
VNORML                         = 000000A3
VSHIFT                         = 000000A2
WCSB0                          = 00000008
WCSB1                          = 00000009
WCSB2                          = 0000000A
WCS_ADDRESS                      ********   X   02
WCS_VALUE                        ********   X   02
WCS_WRITE                        ********   X   02
WRITE                          = 000000CC
WRITE_TAB_PNT                    ********   X   02
WRITE_WR1_32                     00000205 R     02
XD_ADDRS                         ********   X   02
X_CLR_WR0                        ********   X   02
X_CLR_WR1                        00000029 R     02
X_COM_WR0                        ********   X   02
X_COM_WR1                        0000002C R     02
X_MOV_WRWR                       ********   X   02
X_SHIFT_R                        ********   X   02
Y                              = 00000044
YBUSRD                         = 000000EC
ZERO_DAT                         00000054 R     02
ZERO_SHIFT                       0000005A R     02
```

```
                                                         +----------------+
                                                         ! Psect synopsis !
                                                         +----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
. BLANK .                        00000000  (     0.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
CPU2                             00000DDE  ( 3550.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                                      +------------------------+
                                                      ! Performance indicators !
                                                      +------------------------+

Phase                      Page faults   CPU Time       Elapsed Time
-----                      -----------   --------       ------------
Initialization                     23    00:00:00.06    00:00:00.38
Command processing                 34    00:00:00.23    00:00:01.14
Pass 1                           1365    00:00:35.20    00:01:08.90
Symbol table sort                   5    00:00:00.33    00:00:00.83
Pass 2                           1415    00:00:13.27    00:00:27.15
Symbol table output                37    00:00:00.14    00:00:00.50
Psect synopsis output               5    00:00:00.03    00:00:00.03
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             2889    00:00:49.26    00:01:38.93

The working set limit was 1000 pages.
389975 bytes (762 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 222 non-local and 117 local symbols.
4751 source lines were read in Pass 1, producing 52 object records in Pass 2.
151 pages of virtual memory were used to define 150 macros.

                                                     +--------------------------+
                                                     ! Macro library statistics !
                                                     +--------------------------+

Macro library name                                    Macros defined
------------------                                    --------------
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1                                0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBB.LIS/OBJECT=ENKBB.OBJ 8085MAC.MAR+MACDEF.MAC+[]ENKBB.MAC
```